uOttawa

L'Université canadienne
Canada's university

FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTOCTORALES

uOttawa

L'Université canadienne
Canada's university

FACULTY OF GRADUATE AND POSDOCTORAL STUDIES

Soheil Movafagh

AUTEUR DE LA THÈSE / AUTHOR OF THESIS

M.A.Sc. (Civil Engineering)

GRADE / DEGREE

Department of Civil Engineering

FACULTÉ, ÉCOLE, DÉPARTEMENT / FACULTY, SCHOOL, DEPARTMENT

Mapping Three-Dimensional Velocity in a Large Gravel-Bed River

TITRE DE LA THÈSE / TITLE OF THESIS

Dr. C. Rennie

DIRECTEUR (DIRECTRICE) DE LA THÈSE / THESIS SUPERVISOR

CO-DIRECTEUR (CO-DIRECTRICE) DE LA THÈSE / THESIS CO-SUPERVISOR

EXAMINATEURS (EXAMINATRICES) DE LA THÈSE / THESIS EXAMINERS

Dr. Ousmane Seidou

Dr. Andrew Cornett

Dr. Paul Simms

Gary W. Slater

Le Doyen de la Faculté des études supérieures et postdoctorales / Dean of the Faculty of Graduate and Postdoctoral Studies

MAPPING THREE-DIMENSIONAL VELOCITY IN A LARGE GRAVEL-BED RIVER

by

Soheil Movafagh Kerman

A thesis
submitted under the supervision of
Dr. Colin D. Rennie

in partial fulfillment of the
requirements for the degree of
Master of Applied Science
in
Civil Engineering

Department of Civil Engineering
University of Ottawa, Ottawa, Canada, K1N 6N5

August, 2007

We accept this thesis as conforming to the required standard

_____

_____

_____

_____

Your file Votre référence
ISBN: 978-0-494-49253-6
Our file Notre référence
ISBN: 978-0-494-49253-6

ACKNOWLEDGEMENT

I would like to thank my supervisor Dr. Colin Rennie for supervising my studies and for his continued help and support in the course of my research.

ABSTRACT

Three-dimensional images of the boundary, water surface elevation, and velocity vectors in a wandering gravel-bed reach of Fraser River, British Columbia, are prepared. Data were collected through intensive surveying of a 6 km reach using an Acoustic Doppler Current Profiler (ADCP) and Real Time Kinematic GPS (RTK-GPS) mounted on a boat. Nominal channel width was 500 m, and diagonal sections were spaced an average of 120m apart. Using Matlab, binary data in ADCP raw data files are extracted and converted to ASCII format to be usable by Surfer, Tecplot, and other programs. Measured vertical velocities for each single ping ensemble are corrected for boat motion using simultaneous instantaneous change in GPS rover altitude recorded in the navigation data files. Using Surfer, water surface elevation and bottom boundary elevation are interpolated to 25m x 25m grids and combined together to form the boundary of a volume grid. The ASCII files are reformatted in Excel to produce water surface and channel boundary images in Tecplot. The three-dimensional velocity field is interpolated using kriging, and vorticity is calculated based on the velocity field. The error velocity is calculated as well. The high velocity follows the thalweg and reveals the coherence of the interpolated velocity field and the validity of the method. The three-dimensional velocity field and vorticity are analyzed to assess the cause of high bank erosion and sediment transport observed at a particular location. Erosion at this site appears to be related to complicated flow at a channel confluence and 3D vortices produced by flow separation around a riprap nickpoint.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF SYMBOLS

3D	three-dimensional
ADCP	Acoustic Doppler Current Profiler
bottom elev	bottom elevation
c	acoustic pulse traveling speed
D	particle diameter
elev	elevation
error-vel	error velocity
g	gravitational acceleration
GPS	Global Positioning System
n file	navigation file
ρ	density of water
r file	raw data file

surf elev	water surface elevation
S_s	sediment specific gravity
t	acoustic pulse traveling time
τ	bed shear stress
τ^*	dimensionless shear stress
UTC	Coordinated Universal Time
u	velocity in x direction
utme	Universal Transverse Mercator coordinate system-East
utmn	Universal Transverse Mercator coordinate system-North
u_*	shear velocity
v	velocity in y direction
V_b	absolute bed velocity
V_{bt}	bottom tracking measured bed velocity
V_{dgps}	GPS measured boat velocity

velocity-e	east velocity
velocity-n	north velocity
velocity-ver	vertical velocity
w	velocity in z direction
w file	configuration file
ω	vorticity magnitude
ω_x	vorticity in x direction
ω_y	vorticity in y direction
ω_z	vorticity in z direction

1 INTRODUCTION

In river engineering bank erosion is of great importance. Land adjacent to river channels, as well as structures built on the bank of a river are threatened by bank erosion. Aquatic habitat is affected by bank erosion as well. To protect land, structures, and aquatic habitat the cause of erosion and the erosion patterns should be determined, so that methods can be developed to either prevent erosion or to divert its course and change its location.

Erosion and deposition are caused by transportation of sediment. Deposition in a location occurs when the amount of settled sediment in that location is higher than the amount of material that is moved away by flow, and erosion occurs when the amount of the incoming material is less than the amount of outgoing sediment. So, erosion and deposition are functions of the parameters that cause sediment to move. These parameters include channel morphology, bed and bank slope, water depth, discharge, velocity, vorticity, and bank and bed material properties such as grain size, cohesiveness, and vegetation (Rennie and Millar, 2004; Minor 2006).

To analyze the relationship between erosion and these parameters they have to be measured efficiently. A variety of methods are used to measure and calculate parameters such as depth, velocity, vorticity, discharge, slope, and morphology. In this thesis, for the first time these parameters are estimated at high resolution over a large spatial domain. Data collected using both an acoustic Doppler current profiler (ADCP) and a Real Time Kinematic Global Position System (RTK-GPS) are processed with computer programs such as Matlab, Surfer, and Tecplot. Matlab was used for data pre-processing, Surfer was employed to generate gridded boundaries for the channel and water surface, and Tecplot was used both to interpolate the three-dimensional velocity field and to calculate vorticity. The resulting

spatial plots of relevant parameters have previously been unachievable using other methods.

On the bank of Fraser River, British Columbia, immediately downstream of a confluence with Minto Side Channel, where a campground is located, high erosion is taking place. The objective of this study is:

a) to prepare three-dimensional images of the boundary, flow velocity and vorticity in a 5.5 km reach of the river in this zone;

b) to determine the cause of erosion of the eroding bank and therefore to provide a better understanding of the erosion phenomenon.

This thesis is arranged as follows. In Chapter 2 the research previously conducted in this field is reviewed, and the working principles of the instruments used in the study are explained. Chapter 3 introduces the study area, data collection and analysis methods. Chapters 4 and 5 present and discuss the results of the study. Finally, conclusions are made based on the research results.

2 BACKGROUND

2.1 Theory

Sediment is transported in the areas where boundary shear stress is higher than the critical shear stress of the sediment particles (Knighton, 1998). High boundary shear stress occurs where flow is converging and velocity gradient is high (Figure 2-1). Dimensionless shear stress is one of the parameters by which boundary shear stress is specified and is a measure of the tendency of the sediment particles to be transported by the flow (Rennie & Church, 2007; Raudkivi, 1998):

$$\tau^* = \frac{\tau}{\rho g (S_s - 1) D} = \frac{{u_*}^2}{g (S_s - 1) D} \qquad (2.1)$$

where ρ is the density of water, g is gravitational acceleration, S_s is the specific gravity of the sediment material, D is the diameter of sediment particles, and $u_* = \sqrt{\tau/\rho}$ is the shear velocity. In the above formula τ is the bed shear stress, which can be calculated locally using the semi-logarithmic region of the velocity profile gradient $[u = \frac{u_*}{\kappa} ln(h) + \frac{u_*}{\kappa} ln\left(\frac{30}{k_s}\right)]$, where u is the velocity in the bin at elevation h above the mean bed elevation, κ is the von Karman constant (0.41), and k_s is the bed roughness. This assumes locally uniform two-dimensional flow.

Also, wherever turbulence is high, sediment can be moved and erosion occurs. The tendency

of a vector field to rotate can be quantified as an indicator of the magnitude of turbulence that occurs as coherent eddies. Vorticity is a measure of the spatial gradient of velocity that can represent this tendency and is calculated as follows:

$$\omega_x = \frac{\partial w}{\partial y} - \frac{\partial v}{\partial z} \qquad (2.1)$$

$$\omega_y = \frac{\partial u}{\partial z} - \frac{\partial w}{\partial x} \qquad (2.3)$$

$$\omega_x = \frac{\partial v}{\partial x} - \frac{\partial u}{\partial y} \qquad (2.4)$$

where u is the x velocity component, v is y velocity component, and w is the z velocity component. Vorticity in the x direction represents rotation about the x axis.

Vorticity magnitude is calculated as follows:

$$\omega = \sqrt{\omega_x^2 + \omega_y^2 + \omega_z^2} \qquad (2.5)$$

In general, sediment transport is due to interaction between the flow and the boundary and

transfer of energy from the fluid to the sediment through boundary shear stress or vorticity.

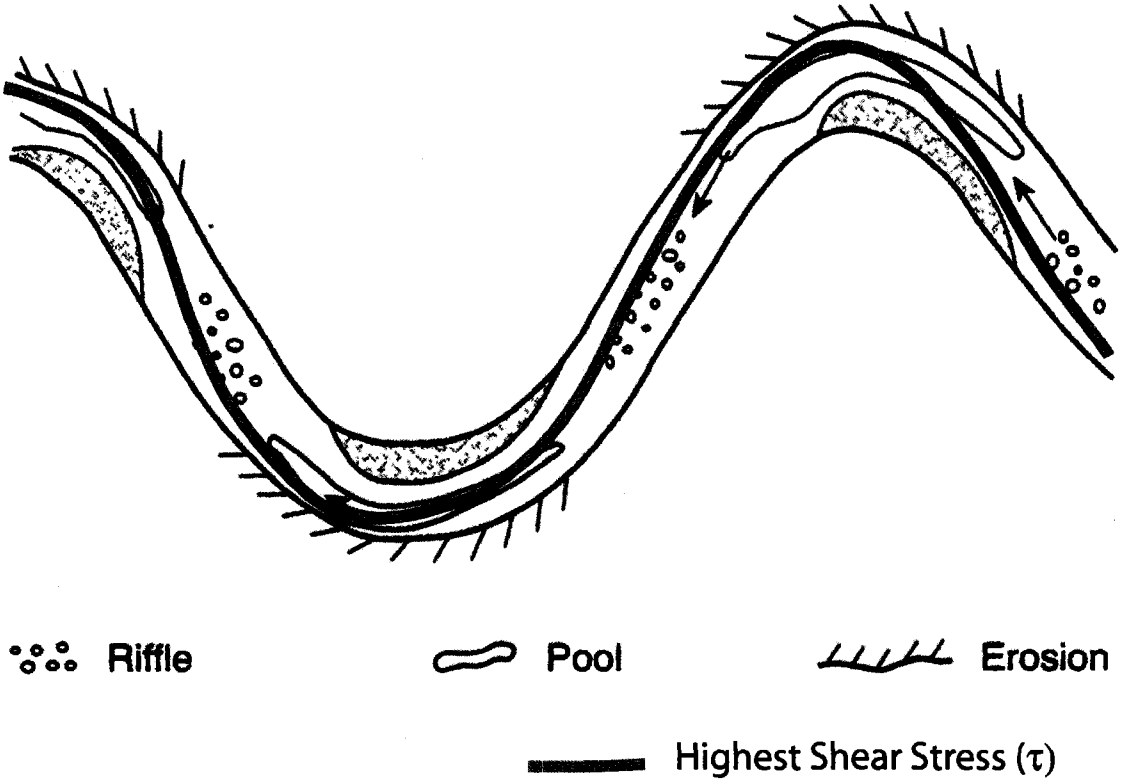

Figure 2-1 Maximum shear stress in a typical meandering river (Rennie, 2006, adapted from Knighton, 1998, based on work of Bill Dietrich).

2.2 Bedload Sampling and Velocity Measurement

Bedload and sediment transport rate are measured by several methods. The most common methods are the conventional sampling method, and measurement of sediment transport rate by an ADCP.

2.2.1 Conventional Method

Physical samplers are used for conventional sampling of bedload. Generally there are two kinds of physical samplers, pressure difference samplers, and slot or pit traps. Pressure difference samplers such as the Helley-Smith sampler (Figure 2-2) are lowered into the flow on the bed. They work by expanding the flow and therefore reducing the pressure at the inlet of the instrument with regard to the outlet. Since the pressure at the outlet of the sampler is equal to the ambient pressure, the inlet pressure will be less than of the surrounding area, and the flow velocity will increase at the inlet. This will trade off the discharge reduction cased by the instrument obstruction of the flow. However, the sampler should be still calibrated. A Helley-Smith sampler usually has a sampling efficiency of 1.0 to 1.5 (Helley and Smith, 1971).

The difference of slot traps (Klingeman and Milhous, 1971) or pit traps (Reid et al, 1980) with pressure difference samplers are that they are permanently installed in the bed of the river and then the amount of the sediment collected in them is measured. The problem they have is that because they are installed in the bed, the spatial variability of the sediment cannot be assessed using them. The advantage of bedload traps (Figures 2-3 and 2-4) is that, for their long sampling time, they give results with low bias and variability in sampling rate. So, they can be used to calibrate other bedload measurement methods and instruments (Bunte et al, 2005).

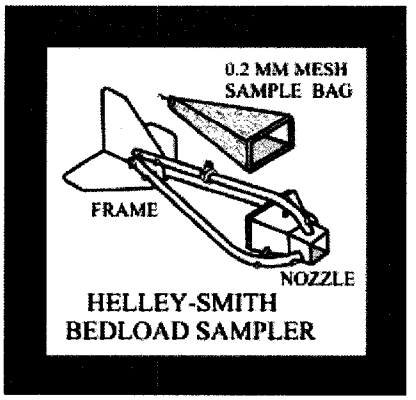

Figure 2-2 Helley -Smith Sampler (USGS, 2003).

Figure 2-3 Bedload trap (Bunte et al. 2005).

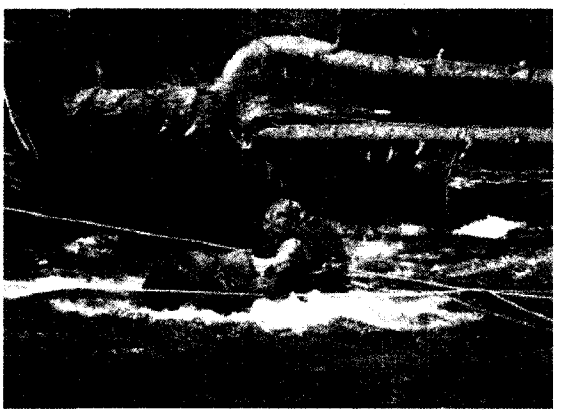

Figure 2-4 Emptying bedload traps (Bunte et al. 2005).

2.3 Acoustic Doppler Current Profiler (ADCP)

2.3.1.1 History

The first ADCPs were made based on the Doppler speed log, which was used for the measurement of the speed of ships through the water or over the sea bottom. The first commercial ADCPs were produced in the mid-1970's by redesigning a Doppler speed log to measure water velocity more accurately in range cells over the measured depth (RD Instruments, 1996).

2.3.1.2 Doppler Effect

An ADCP (Figure 2-5 and 2-6) measures water velocity based on the Doppler effect, i.e. the change in the frequency of the sound wave reflected from a moving obstacle. It emits acoustic beams at a fixed frequency and listens to backscattered sound. The scatterers are particles or plankton floating in the water. By measuring the velocity of these particles, the flow velocity is measured, assuming that the particles move at the same average horizontal

velocity as of the water (RD Instruments, 1996).

Most of the transmitted sound goes forward in all directions, unaffected by the scatterers. However, the small amount that is reflected towards the transducers is Doppler shifted, and enables measurement of the water velocity over the depth profile (RD Instruments, 1996).

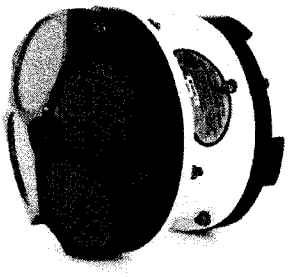

Figure 2-5 An ADCP (RD Instruments, 2005).

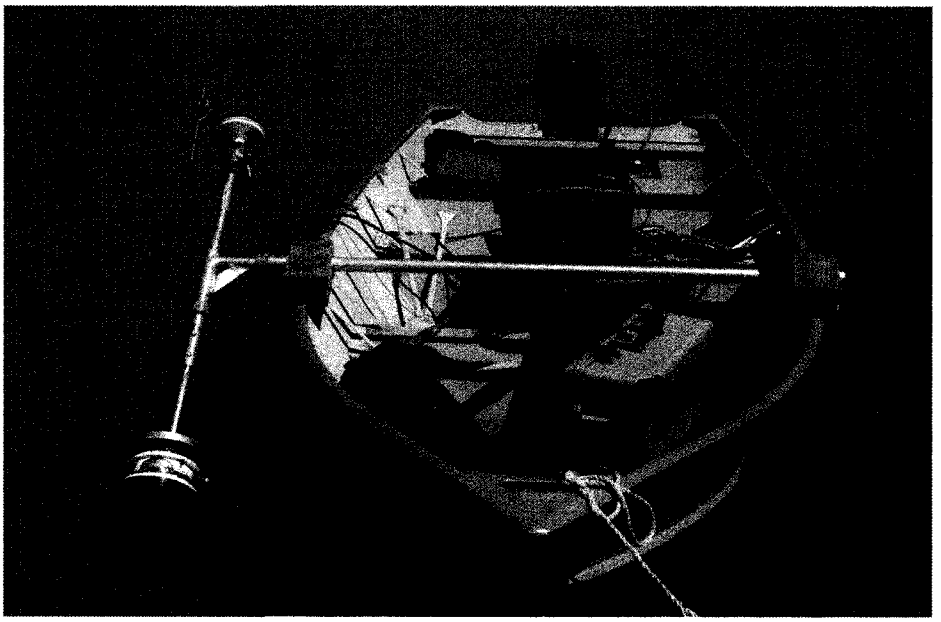

Figure 2-6 ADCP mounted on a boat.

2.3.1.3 Narrowband ADCPs

This kind of ADCP measures the water velocity by sending a relatively long ping and determining the Doppler frequency shift in the backscattered sound. The pings are transmitted independent of each other, and therefore, the echoes are not correlated. Thus, narrowband ADCPs are sometimes referred to as incoherent ADCPs (Brumley et al, 1991; RD Instruments, 1996).

To process the data, the echoes are divided into time gates, to define range cells (Figure 2-7). Each range cell is located a distance of $0.5*c*t$ from the transducer, where c is the speed of sound in water and t is the total traveling time of the pulse. The length of the cells is usually equal to the pulse length. The part of a depth cell that is ensonified by a beam is called a bin. The Doppler shift is determined by finding the spectral peak of the backscattered signals in a range cell (Brumley et al, 1991; RD Instruments, 1996).

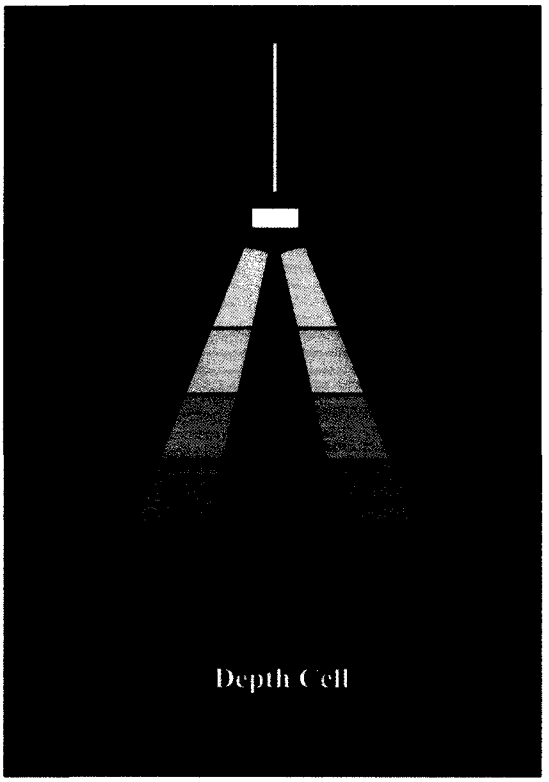

Figure 2-7 ADCP beams, range cells (depth cells), and bins (Rennie, 2006).

2.3.1.4 Broadband Processing

Broadband ADCPs use a series of short pulses, instead of one long pulse. The pulses are correlated. So these ADCPs are also referred to as "pulse to pulse coherent" or for short "coherent" ADCPs. In a pulse coherent ADCP, phase change in the lag between pulses is measured to determine the water velocity, rather than the frequency shift of each pulse. The precision of the measurement can be increased by increasing the lag between the pulses. (Brumley et al, 1991; RD Instruments, 1996).

In a broadband ADCP, the pulses are marked. Therefore, the ADCP can send the pulse at the same time that it is listening to the backscattered pulses, and to listen to backscattered pulses from different emitted pulses, without mixing them together. So, there is no need for the

ADCP to wait for the transmit pulse to propagate to the maximum range before it can send the next pulse. This capacity greatly improves the accuracy of velocity measurement through pulse coherent processing as compared to incoherent processing, while permitting longer ranges as in incoherent processing (Brumley et al, 1991).

2.3.1.5 Phase Measurement and Ambiguity

Narrowband ADCPs directly measure frequency shift. However, broadband ADCPs measure the change in arrival times from successive pulses. This gives a more precise result. However, phase measurement sometimes gives ambiguous results. For example a phase of 120° gives the same results as a 480° phase when it is processed by the electronic circuit of the ADCP (RD Instruments, 1996).

2.3.1.6 Three-dimensional Current Velocity Measurement

An ADCP uses three or four beams to measure velocity components. The radial velocity component (parallel to the beam) is measured in each beam. The beams are emitted in different directions to make it possible to measure different velocity components. However, beams with different directions measure velocity in different places of the flow. So, for a correct ADCP velocity measurement, the flow should be horizontally homogenous. In rivers and oceans the horizontal homogenous assumption is usually a reasonable assumption (RD Instruments, 1996).

In ADCPs with four beams, a redundant vertical velocity is measured, which allows for evaluation of the accuracy of the measured components and the homogeneity of the flow. Otherwise we need only three beams to measure the velocity components of the flow (RD Instruments, 1996). Figure 2-8 shows a single ping ADCP velocity and error profile. A ping or an ensemble is one set of transmission of acoustic pulses by the ADCP transducers and the

data obtained from that.

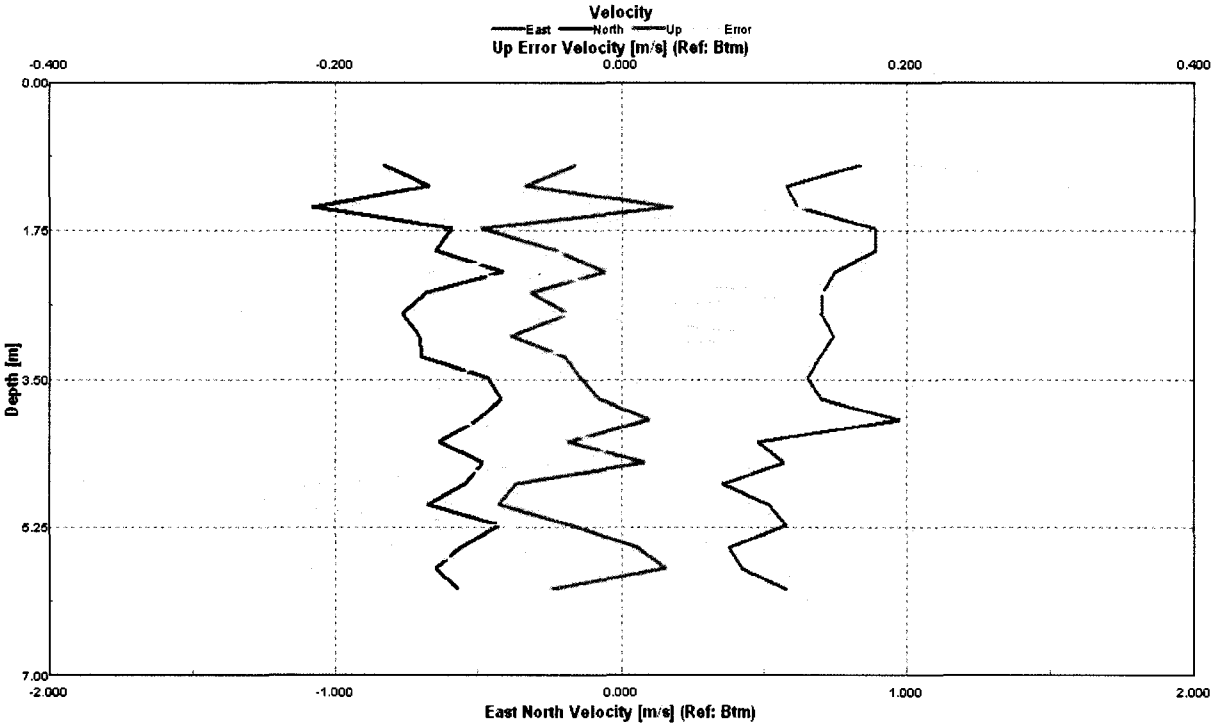

Figure 2-8 An example single ping ADCP velocity and error profile (Rennie, 2006).

One of the parameters by which error is estimated is error velocity and is calculated as follows. The value of water vertical velocity and a component of water horizontal velocity is calculated based on the measurements of two opposite acoustic beams. The same calculation of the vertical velocity and another component of horizontal velocity is made using the other two opposite beams. If the flow is homogeneous, the two measured vertical velocities should be the same. Otherwise the difference between the calculated values of vertical velocities shows the error velocity. The error velocity is scaled to be an indication of the standard deviation of the magnitude of the horizontal velocity (Teledyne Technologies Incorporated, 2007).

2.3.1.7 Bottom Tracking

Bottom tracking is a feature of the ADCP by which the velocity of the bed of the channel can be measured. The difference between bottom tracking and the water velocity measurement feature of an ADCP is that bottom tracking uses longer pulses than those of the velocity measurement feature. The reason is that if a short pulse is used for bottom tracking, the beam cannot sense the bottom with its whole bandwidth when it hits the bottom in a slanting angle, and it produces inaccurate results (Figure 2-9).

The problem with the long pulse is that it can be affected by the suspended sediment close to the bottom, especially when the sediment has a high concentration close to the bed, biasing the measured bed velocity toward the water velocity flow (RD Instruments, 1996).

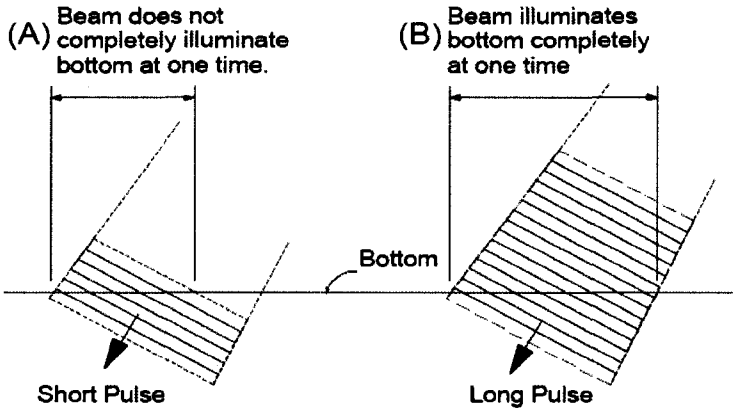

Figure 2-9 A long pulse is needed for the beams to ensonify the entire bottom all at once (RD instruments, 1996).

2.3.1.8 River Morphology, Flow Velocity Profile, Sediment Transport

Dietrich, Smith, and Dunne (1979) researched the relationship between river morphology, boundary shear stress, and sediment transport rate. They found that meandering sand bedded rivers usually have beds that are deformed into bar-pool form and are covered with mobile bedforms in the high flow season. The geometry of the bedforms depends on the flow field through a bend. Migration rates are greater in zones of higher boundary shear stress, and an asymmetric boundary shear stress field develops significant crest obliquity with respect to the average flow direction. Therefore, sediment movement is directed by the bedforms, flow field, and boundary shear stress pattern.

They stated that fluid particles interact with each other and with the channel in two ways, frictionally and inertially. When the internal fluid friction is relatively unimportant and there is little shear stress exchange between fluid particles and they do not rotate significantly, the flow is called inertial flow. However, if shear stress between fluid particles has a much greater influence on the flow field than inertial forces, then the flow is called frictional flow. In the cases that both frictional and inertial forces play roles, if the force balance is generally inertial but the other forces cannot be neglected the flow is called inertially dominated flow. In the same way, if frictional forces dominate the flow, it is called a frictionally dominated flow.

Dietrich, Smith, and Dunne (1979) mentioned that to consider the flow to be frictionally dominated is a practical approach to understand flow and sediment transport in channel bends. However, the effects of inertial forces should be considered for a full understanding of physics of sediment transport in rivers. Topographically induced inertial forces reduce the boundary shear stress in the upstream part of the pool, and increase it in the downstream part of the pool. Therefore, the scour of the pool moves downwards and the high shear stress which is partially responsible for bank erosion is displaced downstream.

They explained that maximum boundary shear stress occurs by the trough-wise current in dunes, and is higher towards the outside of the bend, resulting in sorting of the bed material. In most sand bedded rivers, the direction of bedload transport is developed by the bedforms. Sediment transport and shear stress fields are interdependent terms and interact with each other through changes of bedform orientation. So equilibrium cross sectional geometry of channels is determined by adjustment of the shear stress field and bedforms.

Dietrich and Smith (1983) made some more research about the influence of the bars on flows in curved channels and found out that they cause the flow to be different from what that is observed in a flume especially in the cross-stream flow. The point bar forces the high velocity flow to move towards the pool. It causes an increase of water elevation in the outside band of channel and a decrease in the inside bank. It also causes an increase of boundary shear stress in the outside band and its decrease in the inside bank.

2.4 ADCP Measurement of Bedload Spatial Distribution

Rennie and Millar (2004) were the first who tried to deploy ADCP to make some research in this field and also prepared maps of flow and bedload velocity vectors. They used the bottom tracking feature of the ADCP to measure the bedload transport rate in both gravel-bed reach and sand-bed reaches of the Fraser River. The bedload transport rate was calculated based on the absolute bedload velocity which was calculated based on the following formula:

$$V_b = V_{dgps} - V_{bt} \qquad (2.6)$$

In this formula V_b is the apparent bed velocity, V_{bt} the boat velocity measured by the bottom tracking feature of ADCP, and V_{dgps} the absolute boat velocity measured using a GPS

attached to the boat (Figure 2-10).

They deployed a narrowband ADCP (SonTek Acoustic Doppler Profiler, or ADP) that was connected to a differential GPS using Coast Guard differential corrections, and measured the flow and sediment transport velocity, using a moving boat. This eliminated the problems of anchoring a boat that had to be done in the conventional methods, and enabled them to survey a large area of the reach during a fairly short time. However, it was impossible to gather data in a completely uniform grid, and therefore they needed to interpolate the date onto a uniform grid. The kriging method was used for this purpose. Concurrent Helley-Smith bedload sampling in the sand-bed reach confirmed the trends observed in the prepared maps. In this way they found a coherent pattern in the sand-bed reach of the river and proved that ADCP can be used for measurement of flow and bedload velocity instead of conventional methods (Figure 2-11).

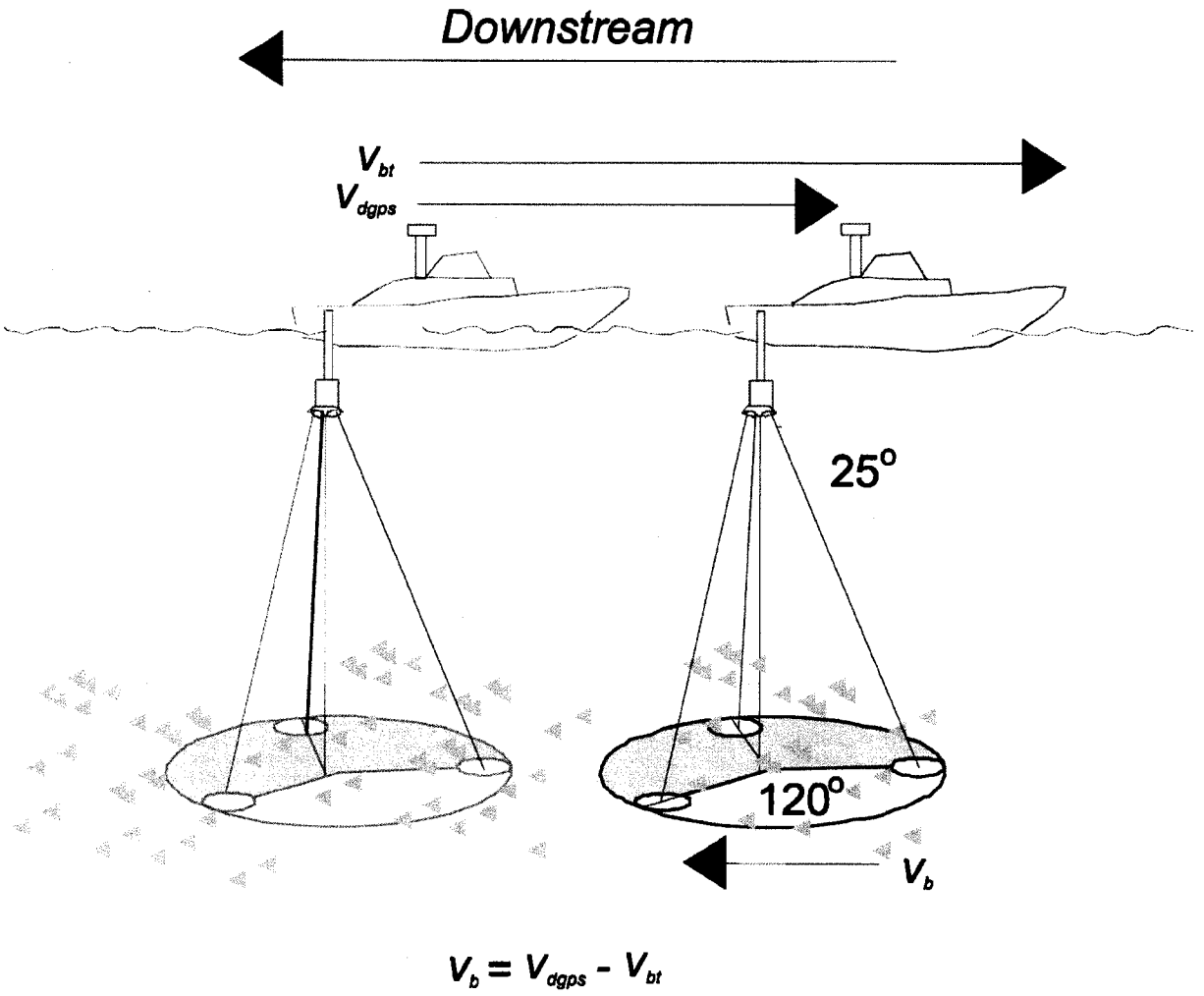

Figure 2-10 Measurement of bedload velocity, using bottom tracking feature of ADCP.

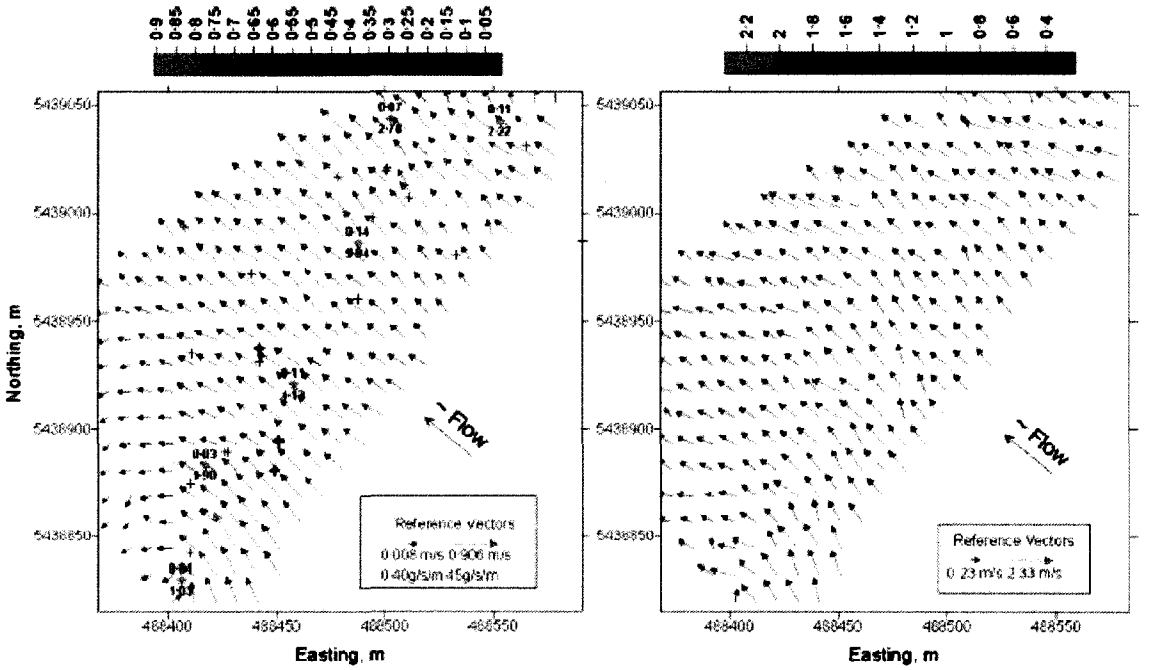

Figure 2-11 (Left) Sand-bed reach kriged bedload velocity distribution in the first hour of data collection. Helley–Smith bedload sample locations marked with +, and bedload samples from second hour of data collection marked by stars with mean bedload velocity (m/s) labeled above and bedload transport rate (g/s/m) labelled below. (Right) Sand-bed reach kriged near-bed water velocity distribution for first hour of data collection (Rennie and Millar 2004).

However, it appeared that the bedload velocity spatial distributions may be biased, probably by near-bed suspended sediment, and that measurement of bedload velocity is more coherent in sand-bedded than gravel-bedded rivers. This was expected because of the sporadic nature of bedload transport in gravel-bed rivers. Sand moves relatively smoothly and evenly, whereas gravel has a more random behaviour (Figure 2-12).

Another problem is that higher bedload velocities were observed in the sand-bed reach during moving boat applications than while the ADP was stationary. They suggested a concurrent sampling of bedload by some other means such as physical sampling or video to calibrate this bias. They also suggested two possible explanations for this discrepancy: either

1) no stationary measurements had been made in a zone of maximum transport rate, or 2) bedload transport was more intense at the time of mobile boat measurements.

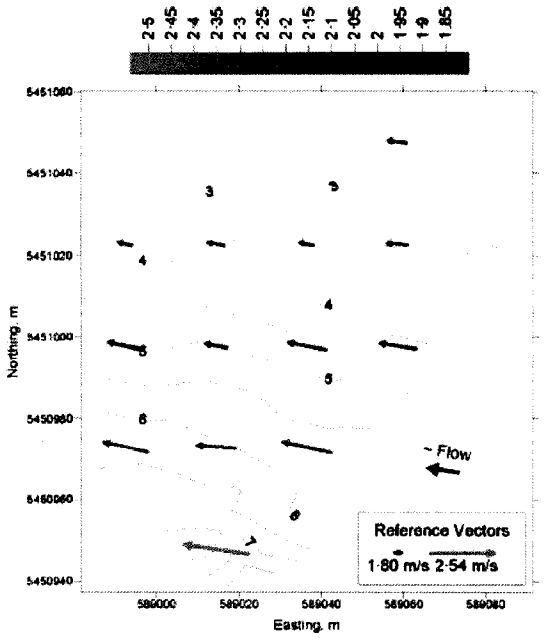

Figure 2-12 Gravel-bed reach spatially block-averaged depth average water velocity (m/s). Centre of vector arrow is centre of block. Overlain on depth (m) contours (Rennie and Millar 2004).

Rennie and Millar (2004) believed that the contribution of the near-bed suspended load in the sand-bed reach was relatively insignificant, whereas it was significant in the gravel-bed reach (Figure 2-13).

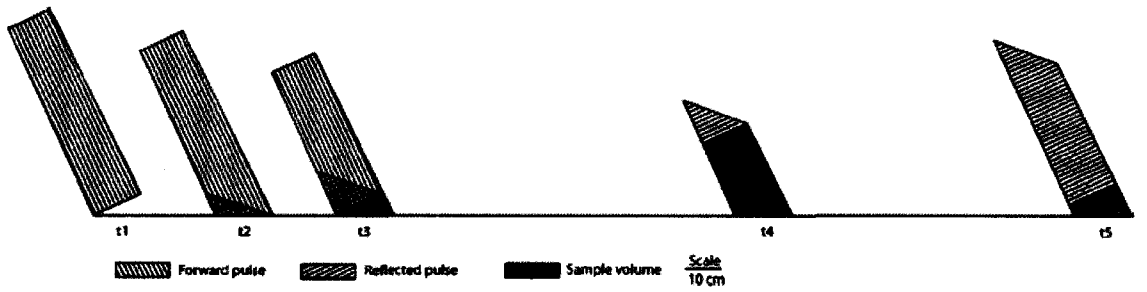

Figure 2-13 Profile of dynamic beam sample volume for a pulse of length 60 cm reflecting off a flat bed 3.05 m below a non-tilted ADP. Beam spread is considered, but the far-field approximation of plane incident and reflected waves is employed. Time is progressing from left to right (scaled 5× for clarity). Sample volume represents locations where suspended scatterers can contribute to the acoustic return, i.e., where the forwarded and reflected pulses are coincident and both the leading and trailing edge contribute to the reflected pulse. Note the sample volume changes dynamically during ping reflection (Rennie and Millar, 2004).

Kostaschuk et al. (2005) further researched use of ADCPs and besides using an ADP to prepare maps of the flow and sediment transport field, tried to determine the suspended backscatterer particle size. Similar to Rennie and Millar (2004), they verified the advantages and disadvantages of using ADCP for this kind of research. They noted the advantages of deployment from a moving boat, using the same instrument for measurement of both flow and sediment velocity non-intrusively, three-dimensional measurement of velocity, and ability to prepare flow and sediment field maps. However, the user should be aware of ADCP limitations , such as sensitivity to the backscatterer particle size, difference between the real ensonified volumes and the point for which the velocity is calculated, coarse measurement of vertical velocity, fragmented bottom tracking records, and a poor understanding of the relationship between the bottom tracking feature of ADCP and the real mechanism of bedload transport. Kostaschuk et al. (2005) emphasized that an ADCP may not yield reliable results for bed velocity in the lee of dunes, where the acoustic waves hit the bed at different depths, especially when the beams get farther apart with the increase of depth.

Kostaschuk et al. (2005) prepared profiles of horizontal and vertical flow velocity (Figure 2-14) and showed that patterns can be observed with regards to the location and form of dunes. The profiles revealed that the horizontal velocities are higher over the crest and lower over the trough of the dunes, which corresponds with the cross section area and expected velocity pattern. They also prepared time series of horizontal flow and bedload velocity over the crest of a large dune (Figure 2-15). The flow velocity revealed a coherent pattern. However, there were fragmented periods in the bottom tracking of bedload velocity. In low velocities the bottom tracking was more coherent, whereas in high velocities they suggested that high sand transport interfered with the bedload velocity measurement.

Kostaschuk et al. (2005) also prepared vertical profiles of velocity and signal amplitude in Lillooet Lake (Figure 2-16) where Lillooet River flowed into the lake in a flood season. Although the signal attenuated with depth, there was a high signal amplitude in the estuary of the river for high sediment concentration.

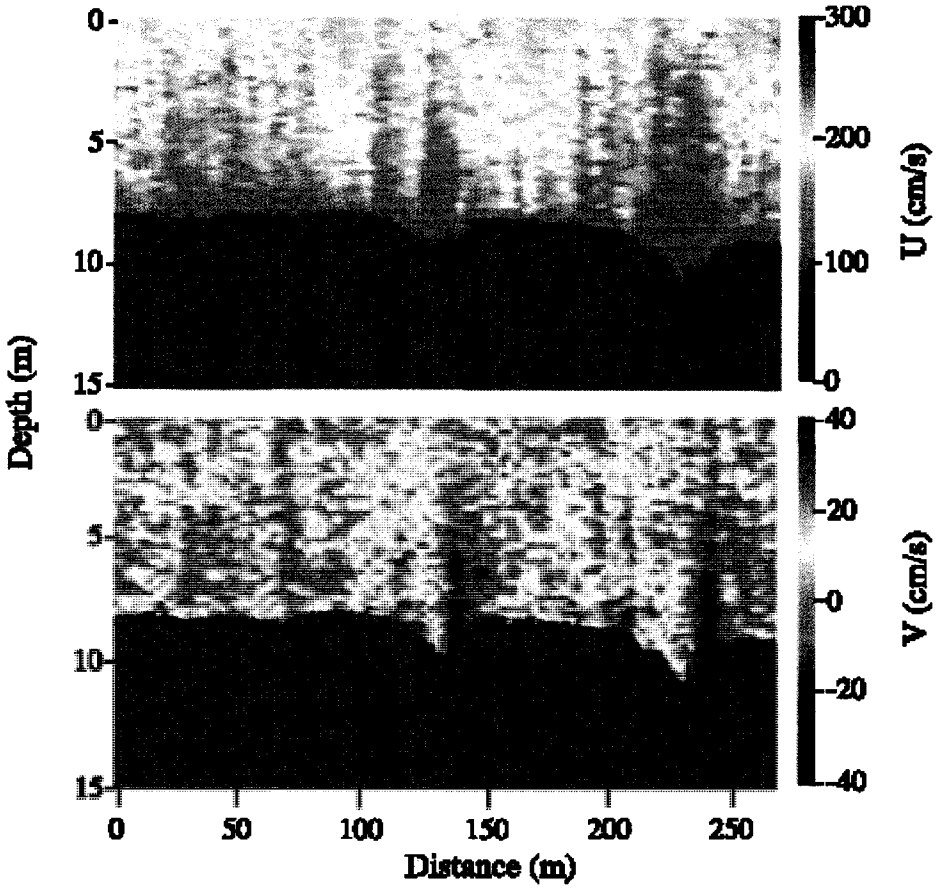

Figure 2-14 Horizontal (U) and vertical (V) velocities measured with the 1500 kHz ADCP deployed from a moving launch in a dune field in the Main Channel of the Fraser Estuary on June 15 1999. Mean reach depth was 12.6m and mean velocity was 2.36 m/s. Mean dune height was 1.92m, mean length was 56m, and mean lower lee slope was 118. Flow is from right to left. The black zones near the bed are contaminated data (Kostaschuk et al. 2005).

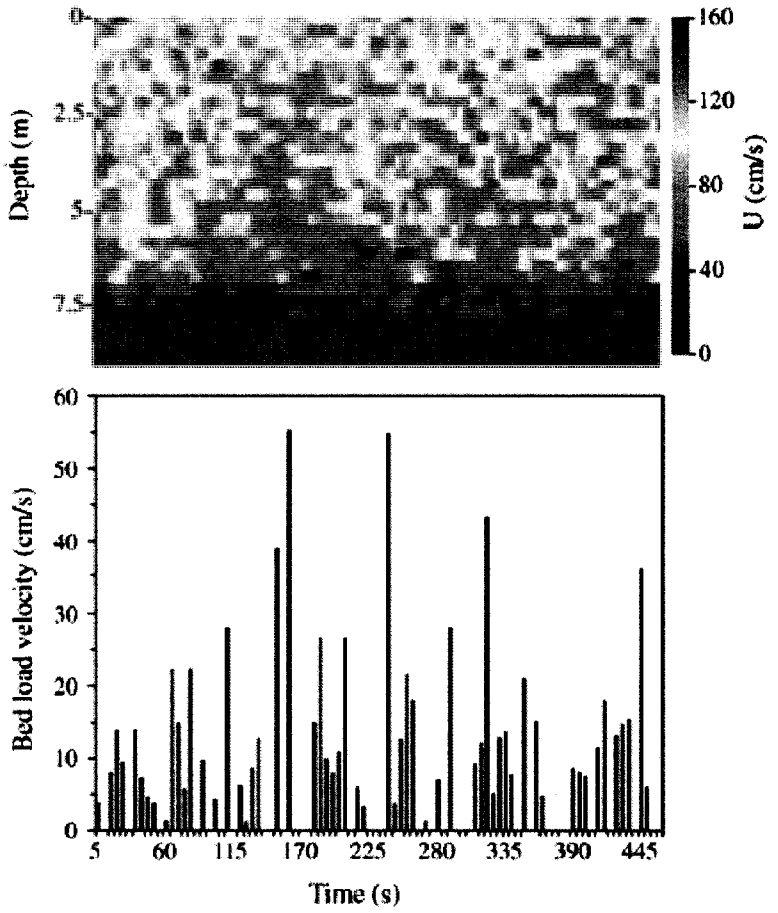

Figure 2-15 Time series of horizontal velocity (U) and bed load velocity measured with the 1500 kHz ADCP from a launch anchored over a dune crest in Canoe Pass on June 3, 2000 (Kostaschuk et al. 2005).

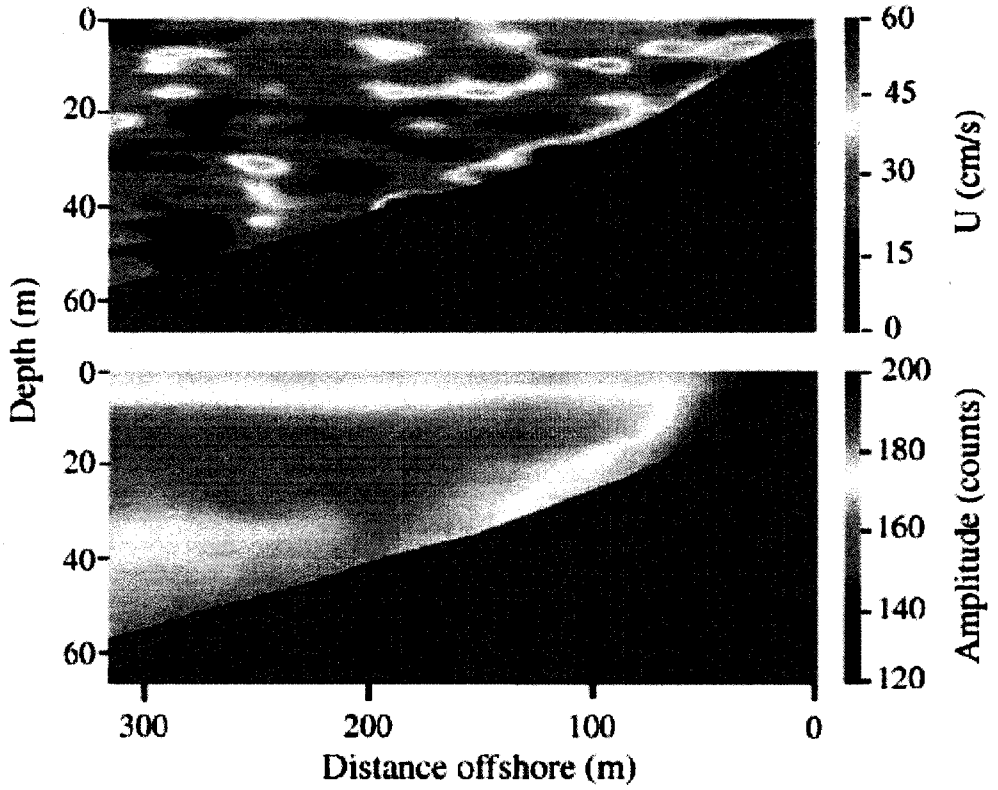

Figure 2-16 Horizontal velocity (U) and signal amplitude measured with the 500 kHz ADCP deployed from a moving launch in Lillooet Lake on August 22, 2001 (Kostaschuk et al. 2005).

Gaeuman and Jacobson (2006) researched the sand bed Missouri River using a 600 kHz broadband ADCP and noticed that up to threshold transport stage, the measured bedload velocity increases almost linearly with the increase of flow velocity and sediment transport rate. However, for the higher velocities, measured bedload velocity and the scatterer concentration increases sharply (Figure 2-17). Gaeuman and Jacobson (2006) developed a physically-based model to relate bed velocity and the mean velocity of particles moving close to the bed. This model is a function of two parameters. The first parameter is the strength of the backscattered signal from the moving bedload relative to the strength of the total backscattered signal from both stationary and moving bed, and the second parameter deals with the spatial variability in the near-bed sediment transport field and with the changes

of transport with changes in bedform morphology.

Gaeuman and Jacobson (2006) found that in the presence of smaller bedforms, with flow separation, the bedload velocity is near zero in large areas of the bed. However, with the increase of the average flow velocity and appearance of larger bedforms, the bedform velocity becomes more evenly distributed and there will be fewer areas with zero bedload velocity.

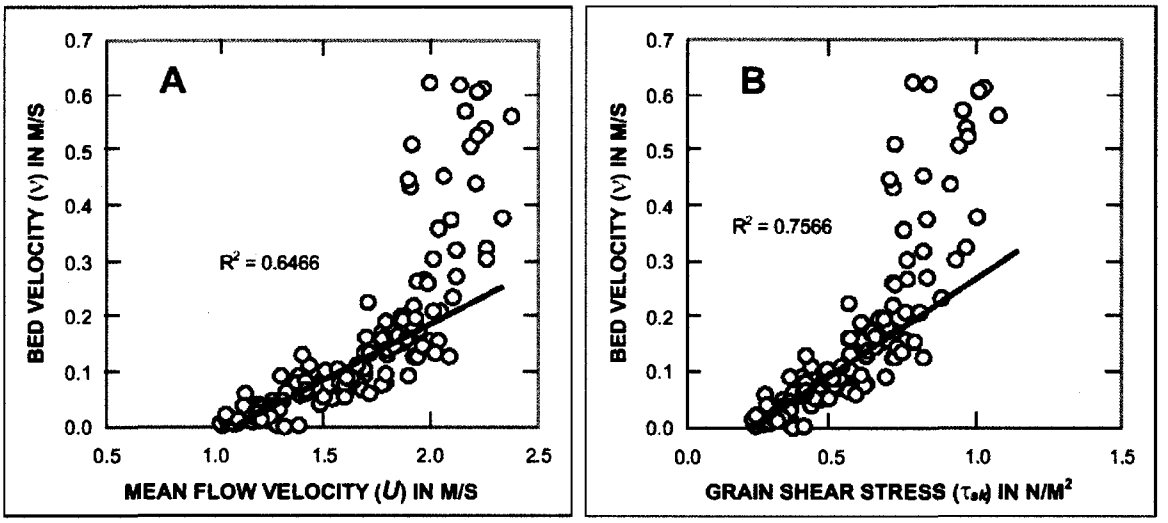

Figure 2-17 Graphs showing bed velocity plotted against A) depth-averaged flow velocity and B) grain shear stress (Gaeuman and Jacobson, 2006).

Rennie and Church (2007) surveyed a 5.5 km wondering gravel-bed reach of the Fraser River to prepare maps of spatial distributions of depth averaged water velocity, shear velocity, and apparent bedload velocity. They used an ADCP and RTK-GPS to survey the river by a boat (Figure 2-18). The maps were prepared by interpolating data using the kriging method.

The maps showed coherence between flow depth, depth averaged water velocity, shear velocity, and apparent bedload velocity (Figures 2-19 to 2-22). The high water velocity, shear

stress, and bedload transport followed the thalweg of the reach. The maximum shear stress occurred immediately downstream of the pools, where flow converged against the channel bank, as expected for flow in meandering channels. Low velocities and shear stress occurred on shallow point bars and in locations of flow separation.

Rennie and Church (2007) suggested that surveying the river by combination of ADCP and GPS, mounted on a boat, could produce reliable spatial distributions in large scales in a relatively short time. They also suggested that the deep pool and rapid bank erosion which occurs downstream of the confluence of Fraser River with Minto Side Channel is due to the turbulence and complex three-dimensional flow caused by the confluence with Minto Side Channel and armouring of the bank upstream and downstream of the location of bank retreat, and not by high boundary shear stress which is generally believed to be the cause of sediment movement. High boundary shear stress occurs along the thalweg and where the flow is converging. However, this pool is located upstream of the thalweg, and the water velocity and the boundary shear stress are relatively low. Rennie and Church (2007) suggested that three-dimensional images of the flow velocity field be prepared for analysis of the cause of high erosion at this site. The present research fulfills this objective.

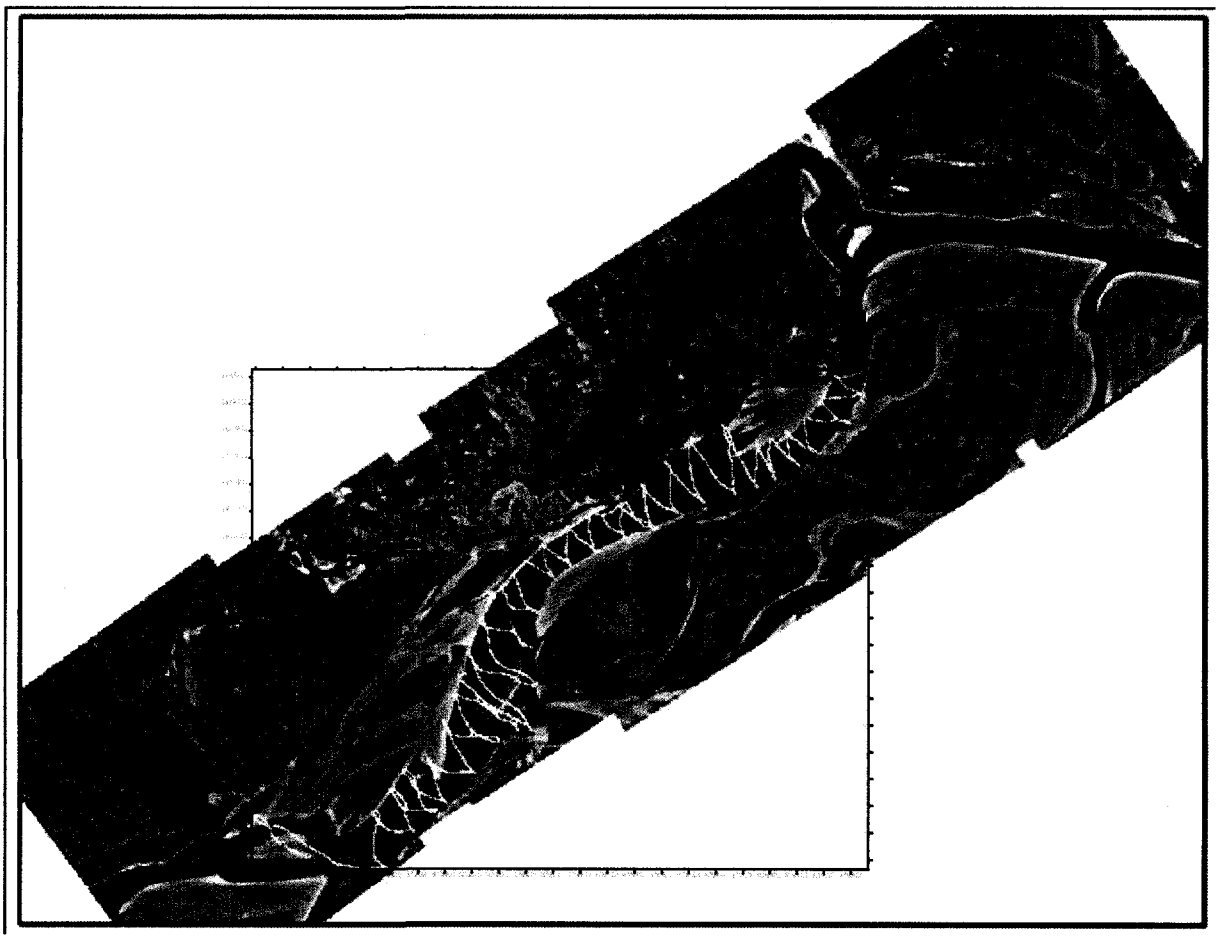

Figure 2-18 ADCP transects (yellow) collected June 24 and 25, 2006 in an approximately 6 km long study reach. Flow is from NE to SW. Harrison River enters from the north. Orthorectified air photo image taken at low flow in early spring 2006 (Rennie and Church, 2007).

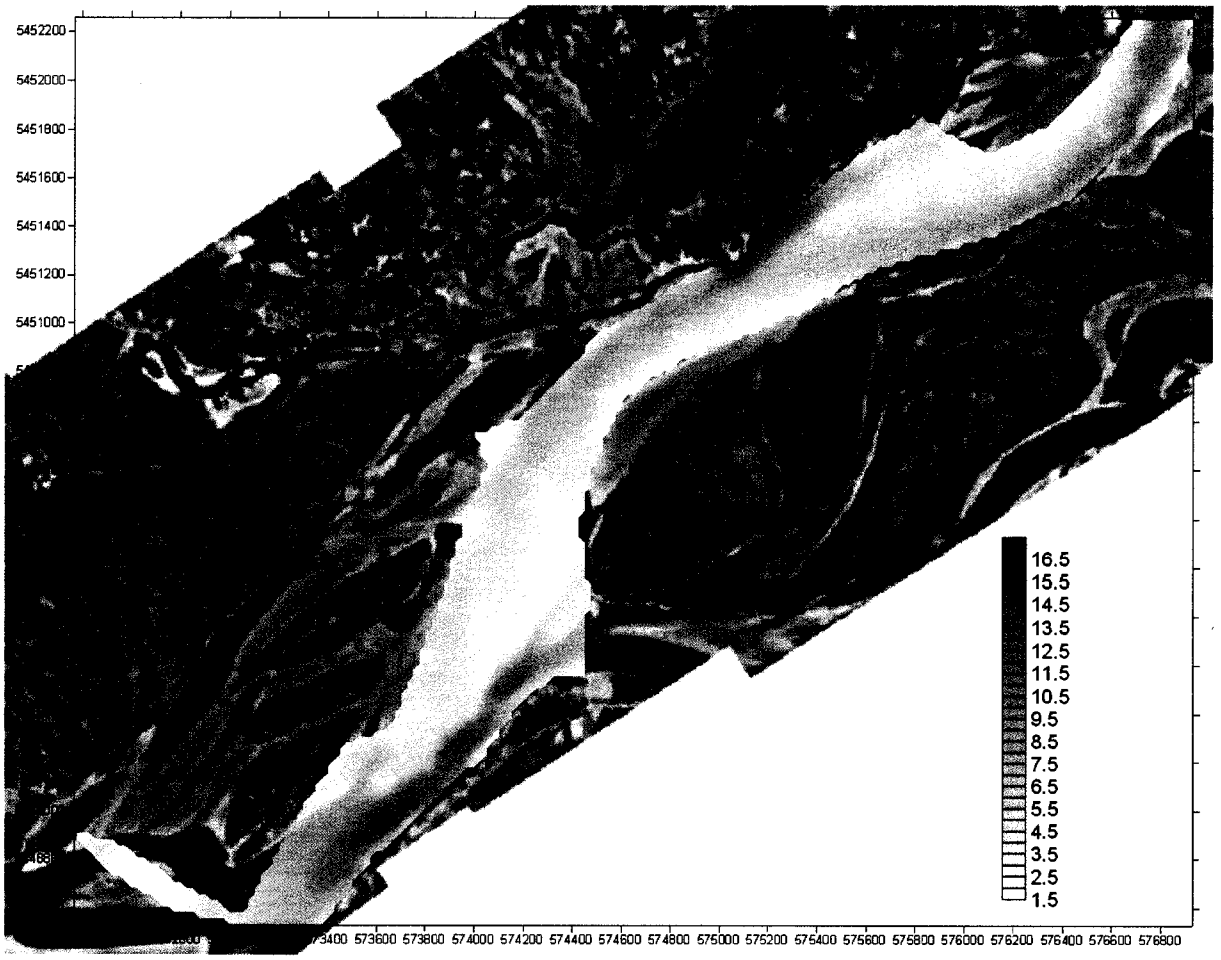

Figure 2-19 Interpolated flow depths (m), June 24-25, 2006. Contour interval 0.5 m (Rennie and Church, 2007).

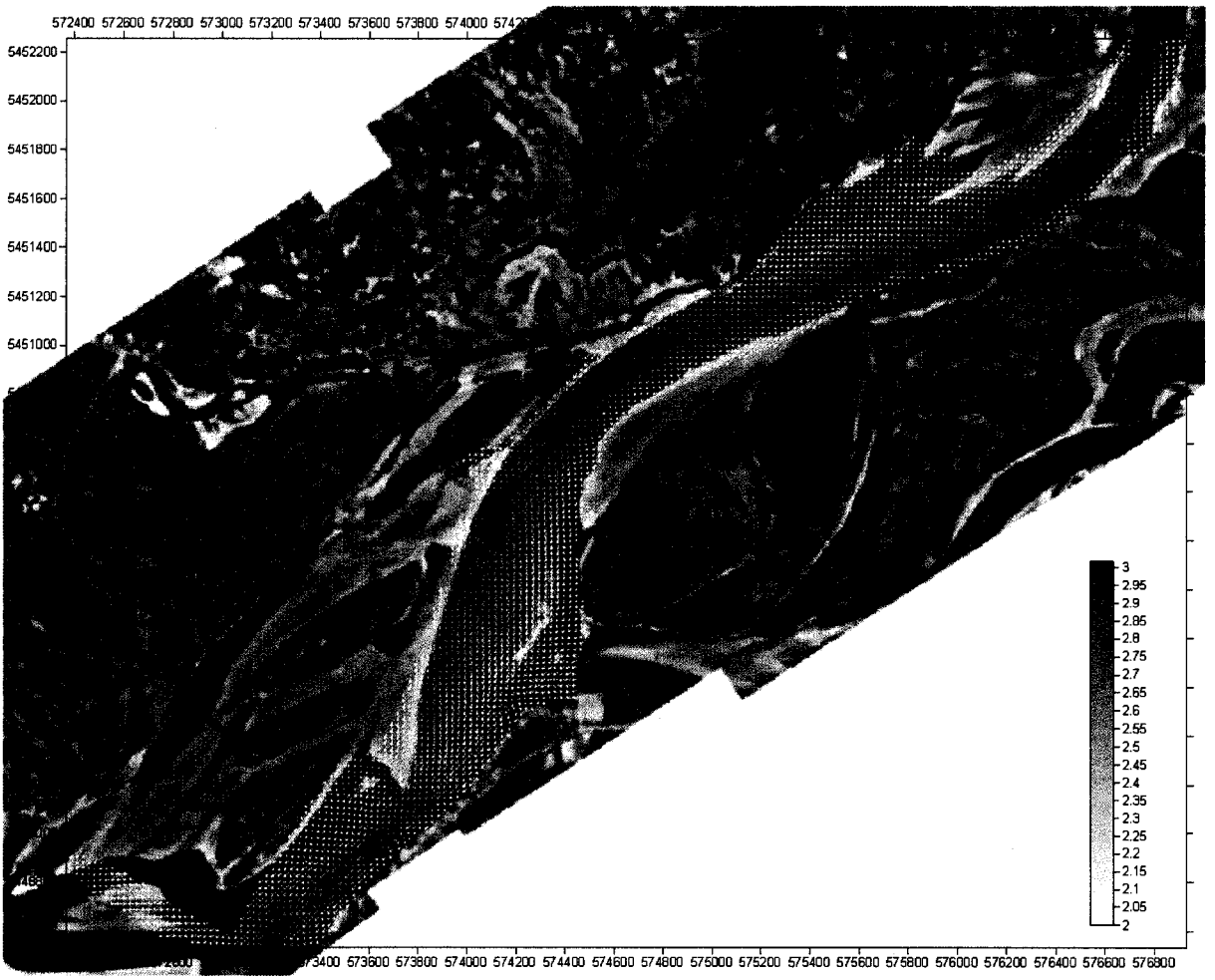

Figure 2-20 Interpolated depth averaged velocity (m/s), June 24-25, 2006 (Rennie and Church, 2007).

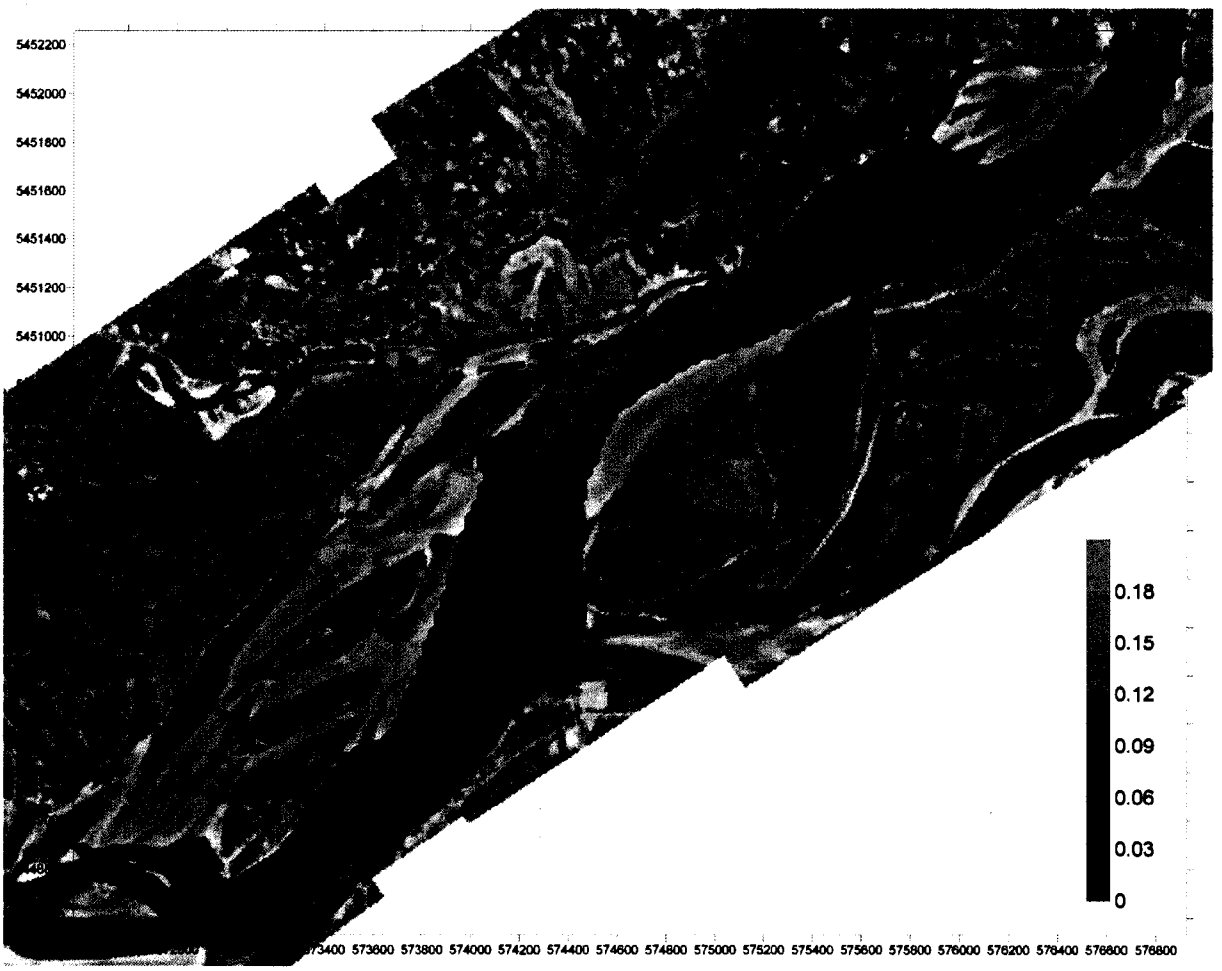

Figure 2-21 Interpolated shear velocity (m/s), June 24-25, 2006 (Rennie and Church, 2007).

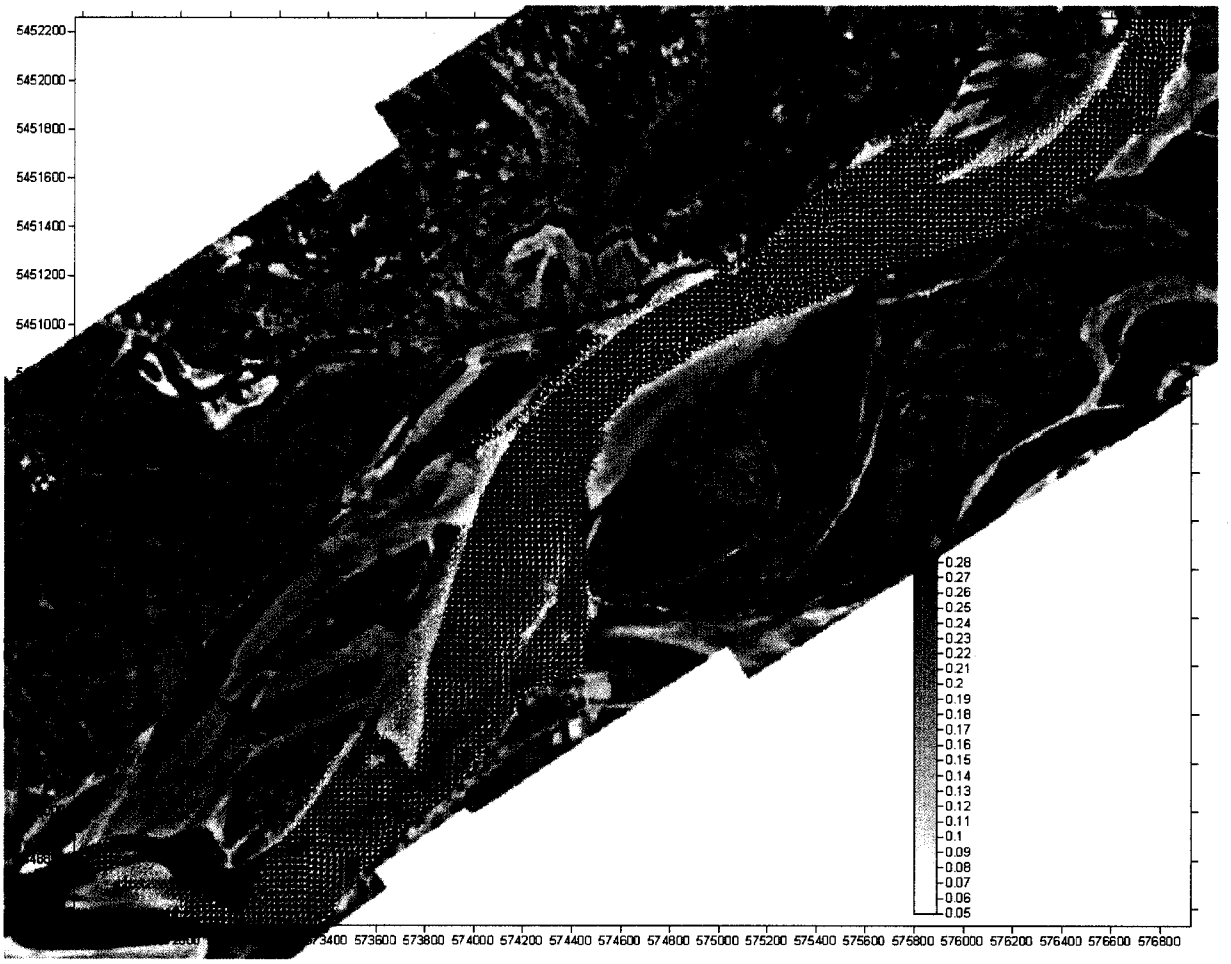

Figure 2-22 Interpolated bedload velocity (m/s), June 24-25, 2006 (Rennie and Church, 2007).

2.5 Combining RTK-GPS with ADCP Measurements

A global positioning system (GPS) determines the position of the instrument through communication with at least four of the satellites that are in orbit around the globe. It determines the position by calculating the distance between the instrument and the satellites and finding the unique point in the space which has the respective distances from the satellites (Rennie and Rainville, 2006).

Over the last several decades GPSs with higher accuracy have been developed. The most sophisticated kind of GPS is Real Time Kinematic GPS (RTK-GPS). Every RTK-GPS consists of two stations, the Rover station and the Base station (Figures 2-23 to 2-25). The Rover station, which consists of an antenna, a receiver, and a radio is installed where the position should be measured such as on a boat. The location of the Rover antenna is what is measured. The Base station also consists of an antenna, a receiver, and a radio. However, it is installed on a stationary position such as the bank of the river, where it can easily communicate through its radio with the Rover radio. Both Rover and Base should also be in positions where they can communicate with at least four of the twenty four satellites around the earth to measure accurate position. In good working conditions, when the communication between the antennas and satellites are made properly, RTK-GPS measures positions within an accuracy of 1cm, which is a very high accuracy (NovAtel 2005; Rennie and Rainville, 2006).

GPS receivers measure the location by measuring the time taken for the satellite electromagnetic wave to reach the GPS antenna. By multiplying this time by the electromagnetic wave speed, the distance between the satellite and the GPS antenna is calculated. Since the location of every satellite is known precisely at every moment, the location of the GPS antenna at every moment can be calculated. The unique point in the space which has the respective calculated distances from the four satellites is the GPS antenna position (NovAtel 2005; Rennie and Rainville, 2006).

One reason for the high accuracy of RTK-GPS is that it measures the phase change of the electromagnetic wave to measure its traveling time, whereas lower quality GPSs use phase change in the coded signal, which yields much lower resolution. Another reason for high accuracy of RTK-GPS is that it eliminates the error caused by the interference of atmosphere layers in the antenna-satellite communications. If the atmosphere layers interfere with the electromagnetic waves communicated between satellites and GPS antennas and change their velocity, then there will be error in the measured position, unless this error is calculated and

eliminated. The Base is designed for this purpose. It is located in a stationary position and its location is known. Any temporal changes in the velocity of the electromagnetic waves will cause an apparent change in the Base location, which gives the error. Since the Rover and the Base are located a short baseline distance from each other relative to their distances from the satellites, any atmosphere interference will have the same impact on their communication with satellites, and both will have the same amount of error. So, the real position of the Rover is calculated by subtracting the error calculated by the Base (Rennie and Rainville, 2006).

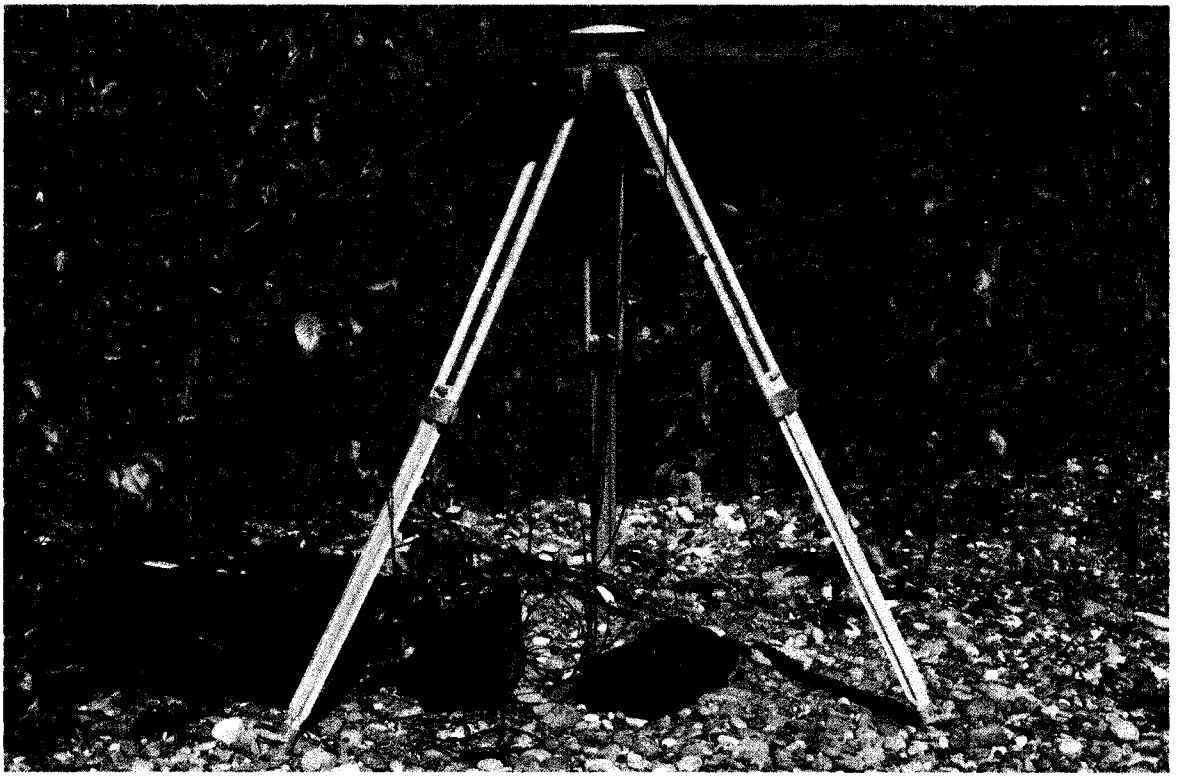

Figure 2-23 RTK-GPS Base; the antenna and Base radio are installed on the top and side of the tripod. The Base receiver is in the black case. A 12V battery provides the power for the system.

Figure 2-24 The ADCP and two GPS Rover antennas on the boat. When surveying, the mount has to be rotated 90° so the ADCP goes down into the water and the GPS antennas stand up towards the sky.

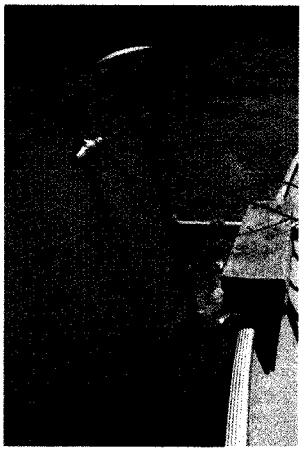

Figure 2-25 The GPS Rover antennas. This picture also shows the mount position when river is being surveyed.

3 METHODS

3.1 Study Area

The study area is a 5.5 km graved-bed reach of the Fraser River, British Columbia, beside the city of Chilliwack (Figure 3-1). The beginning point is the confluence of the Fraser River with the Harrison River, 576800 utme 5452200 utmn, and the end point is the downstream end of Queens Island, 572400 utme 5448600 utmn. The symbols utme and utmn represent Universal Transverse Mercator coordinates (m) in east and north directions, respectively. The average width of the River is 600 m and mean annual flood is 8000 m^3/s (Figure 3-2). Immediately downstream of the confluence of the Fraser River and the Harrison River, on the right bank, the Harrison Knob and the Harrison Hill are located (Figures 3-3 and 3-4). Lower down in the reach, close to the right bank, the Queens Bar and the Queens Island are located (Figures 3-5 and 3-6). Almost in the middle of the reach, close to the left bank, the Minto Island is located. Between Minto Island and the left bank, the Minto Side Channel is located. The Minto Side Channel branches off the Fraser River upstream of Minto Island and then joins the Fraser River after the Island . At the bank of the Fraser River, downstream of its confluence with the inflowing Minto Side Channel, a campground is located. This location is marked by camping symbol on Figure 3-1. The bank is made of silt on top of coarse material. At the confluence junction the bank is reinforced briefly by rip rap to stop erosion. However, the bank of the campground is eroding and retreating after this nickpoint (Figures 3-7 to 3-18).

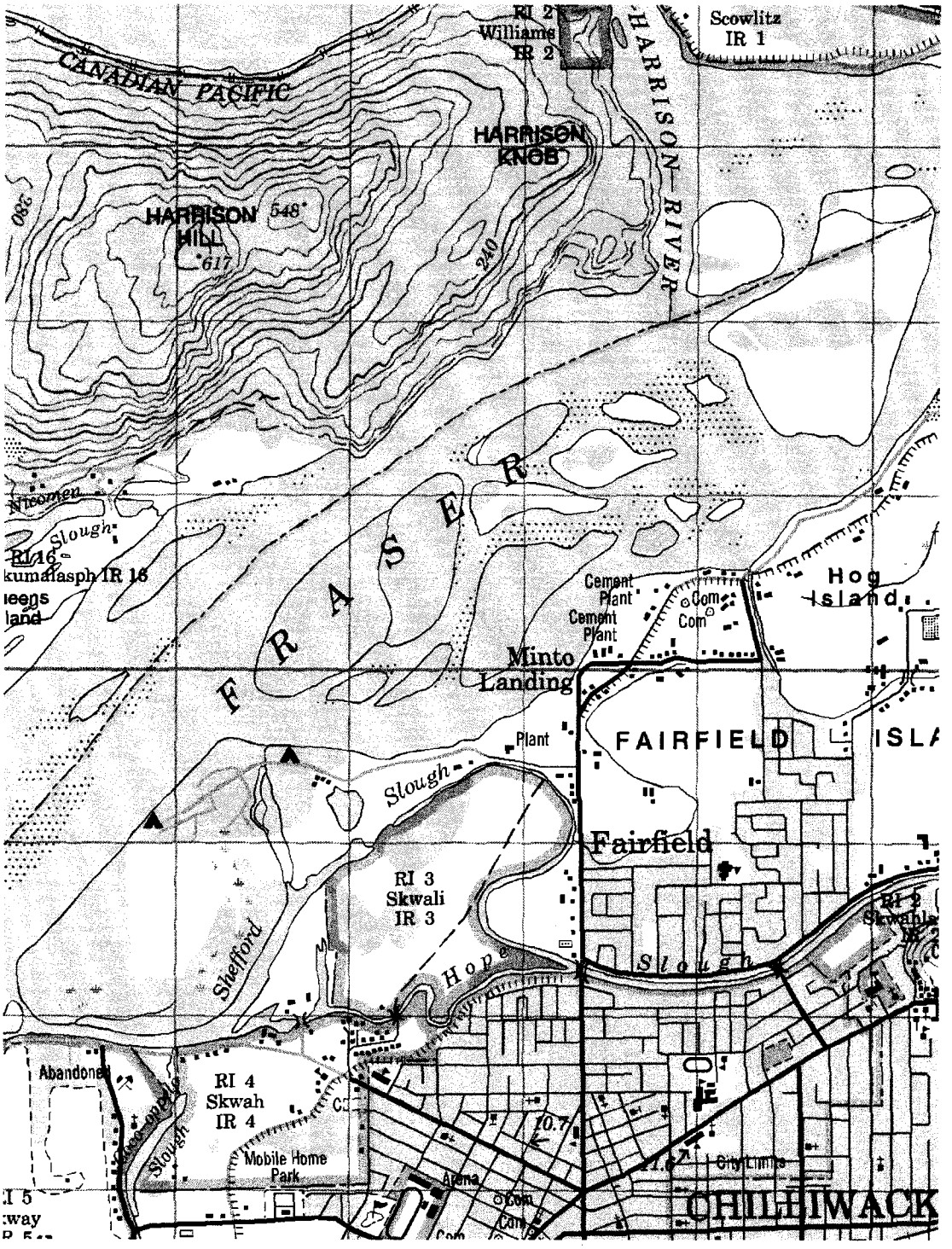

Figure 3-1 Map of the study area (Natural Resources Canada, 1999).

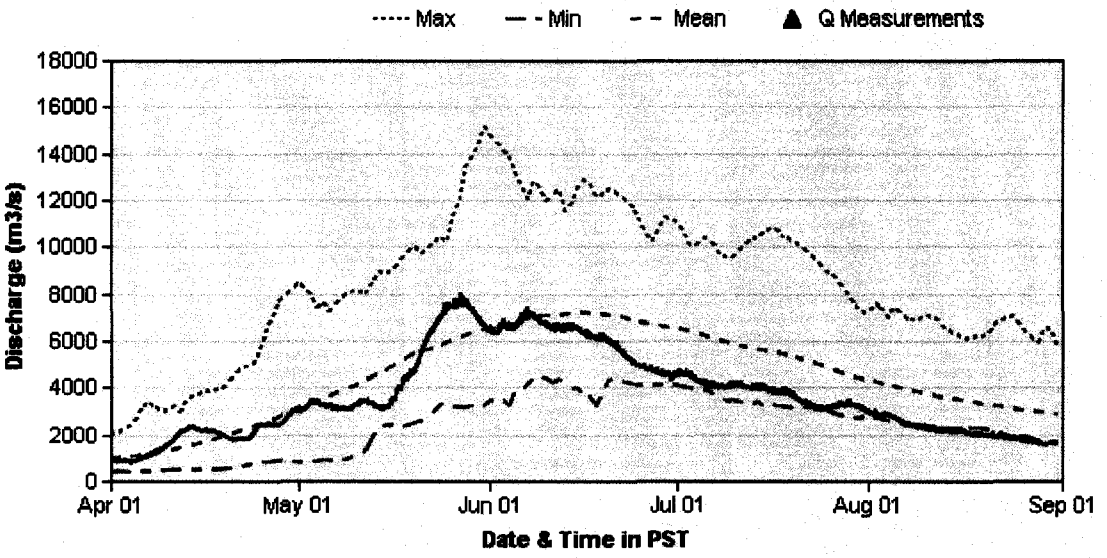

Figure 3-2 The Fraser River hydrograph for the 2006 freshet, upstream of study area (Water Survey of Canada, Fraser River at Hope gage station 08MH024).

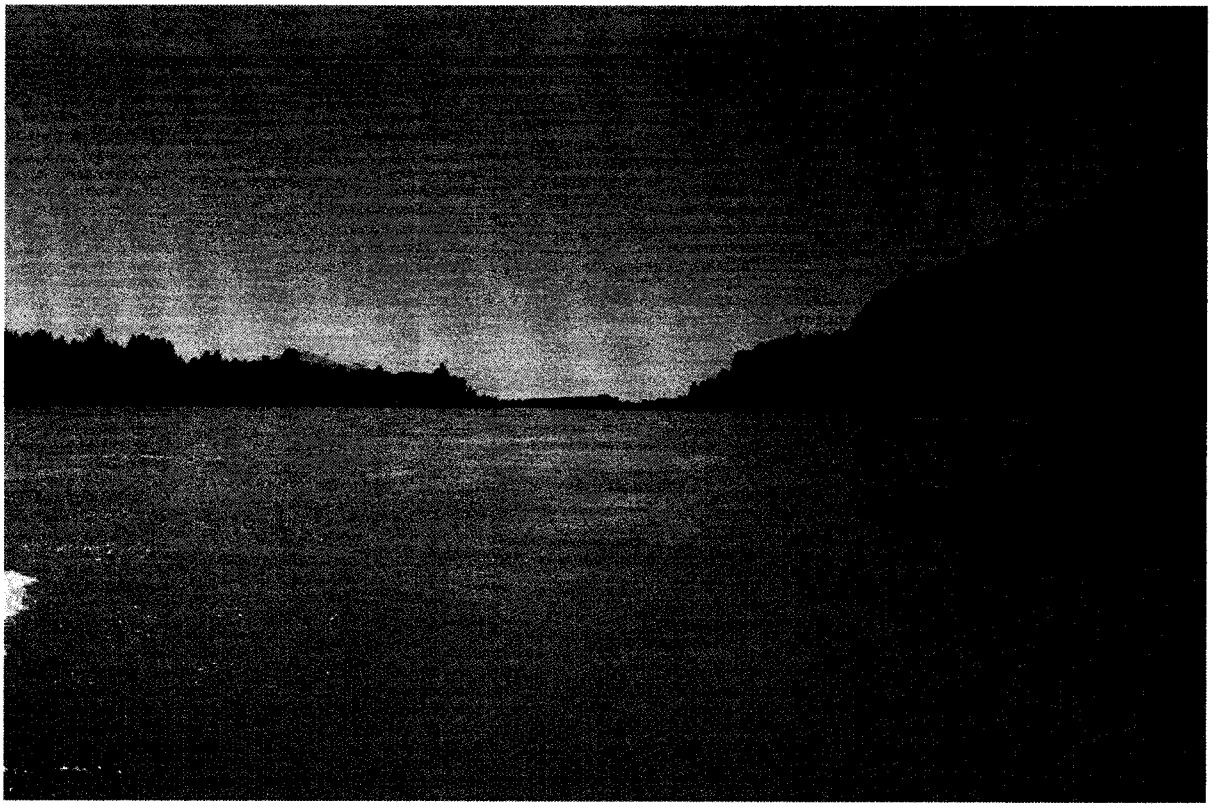

Figure 3-3 Picture of the Fraser river looking towards downstream. The Harrison Knob can be seen on the right hand.

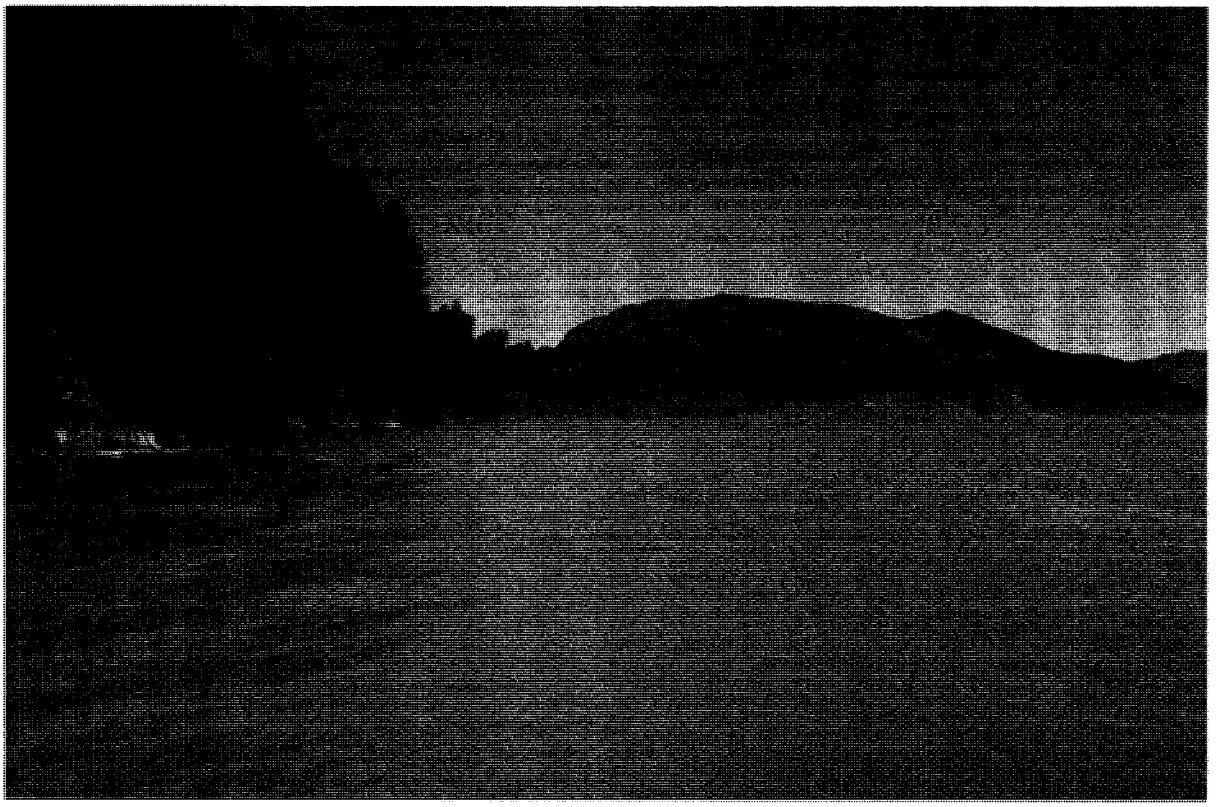

Figure 3-4 Picture of the Fraser river looking towards upstream. The Harrison Hill can be seen on the left hand.

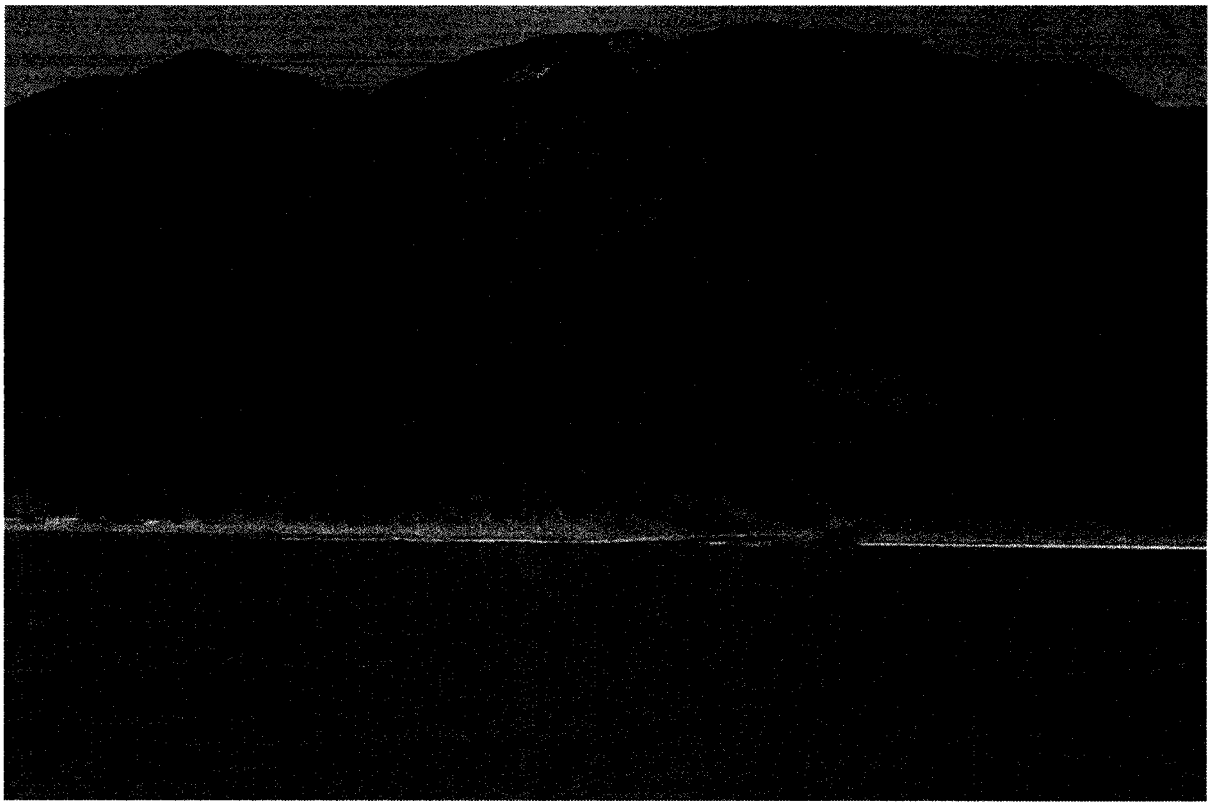

Figure 3-5 The Queens Bar (in front).

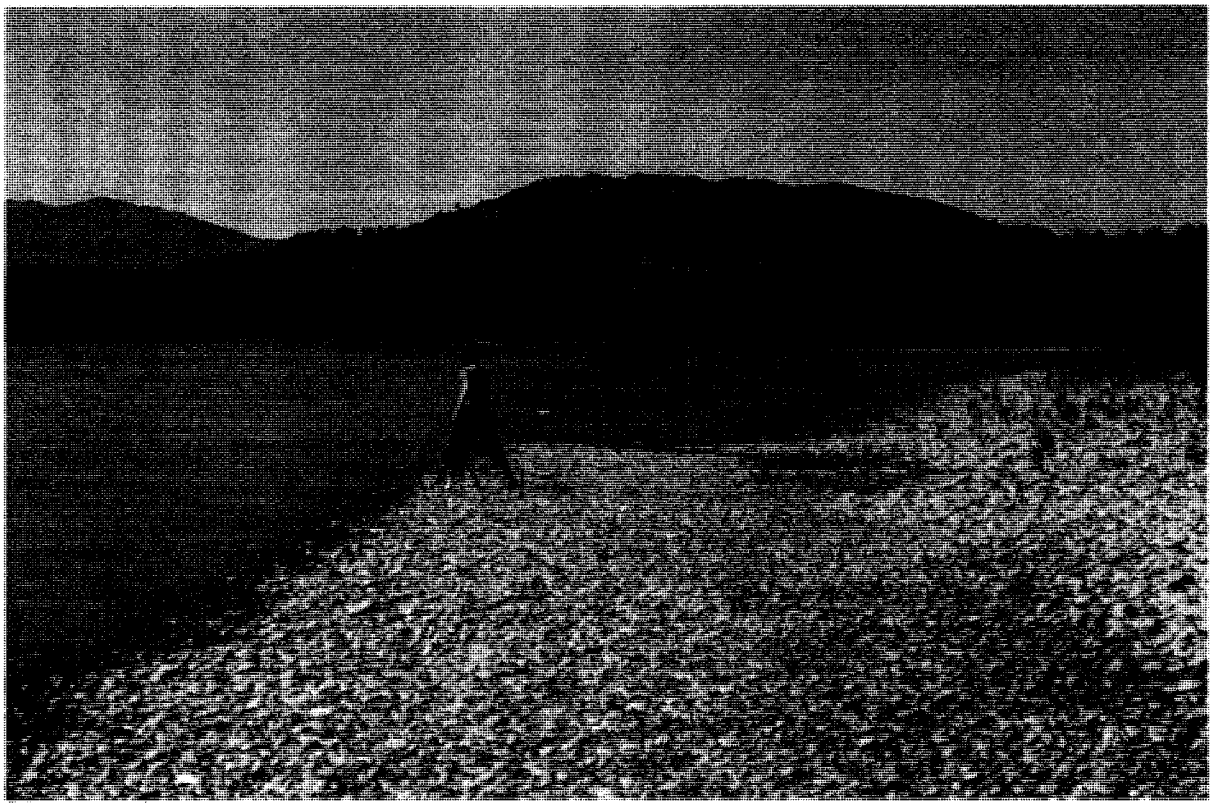

Figure 3-6 The picture of the Queens Island taken from the Queens Bar.

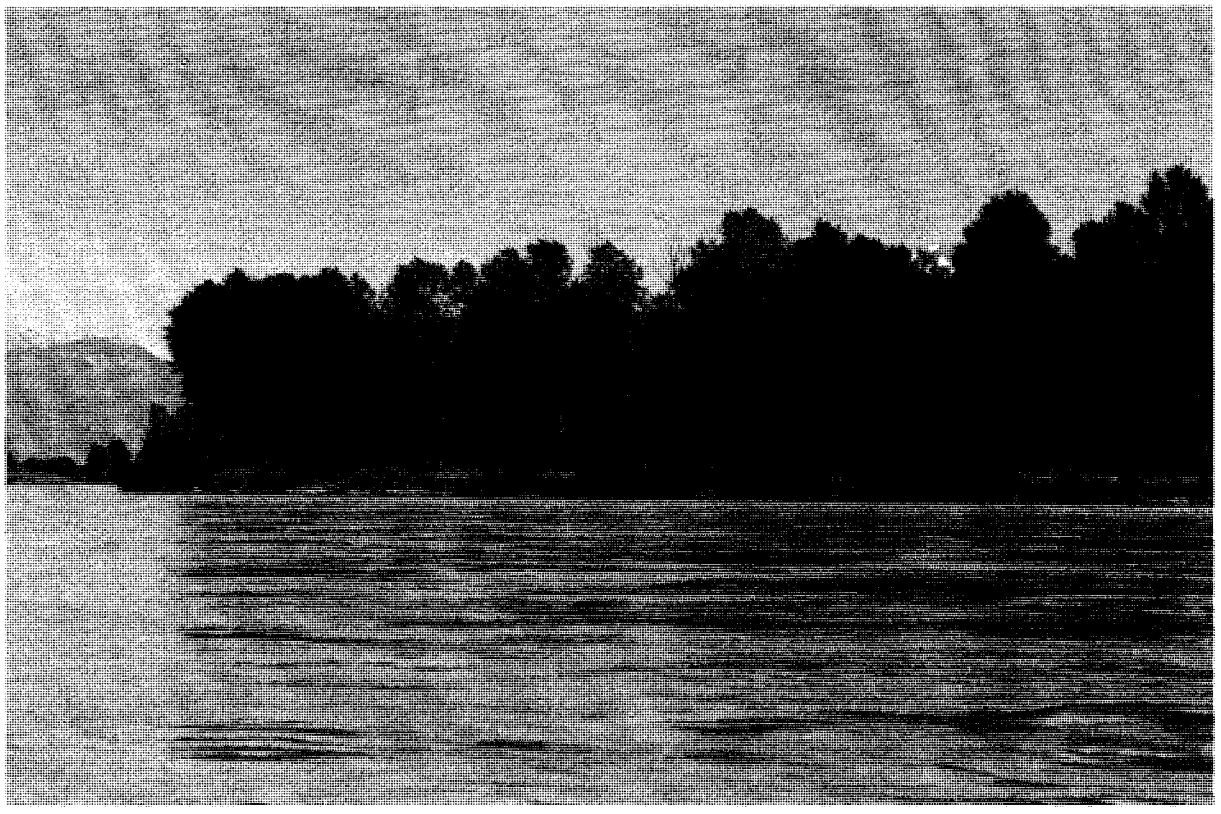

Figure 3-7 Figure 3-8 The campground bank at the confluence of the Fraser River and the Minot Side Channel taken on the Fraser River. The Minto Side Channel can be seen on the left side of the picture. The eroding cutbank and the armouring rip-rap on its left side can also be seen in the picture.

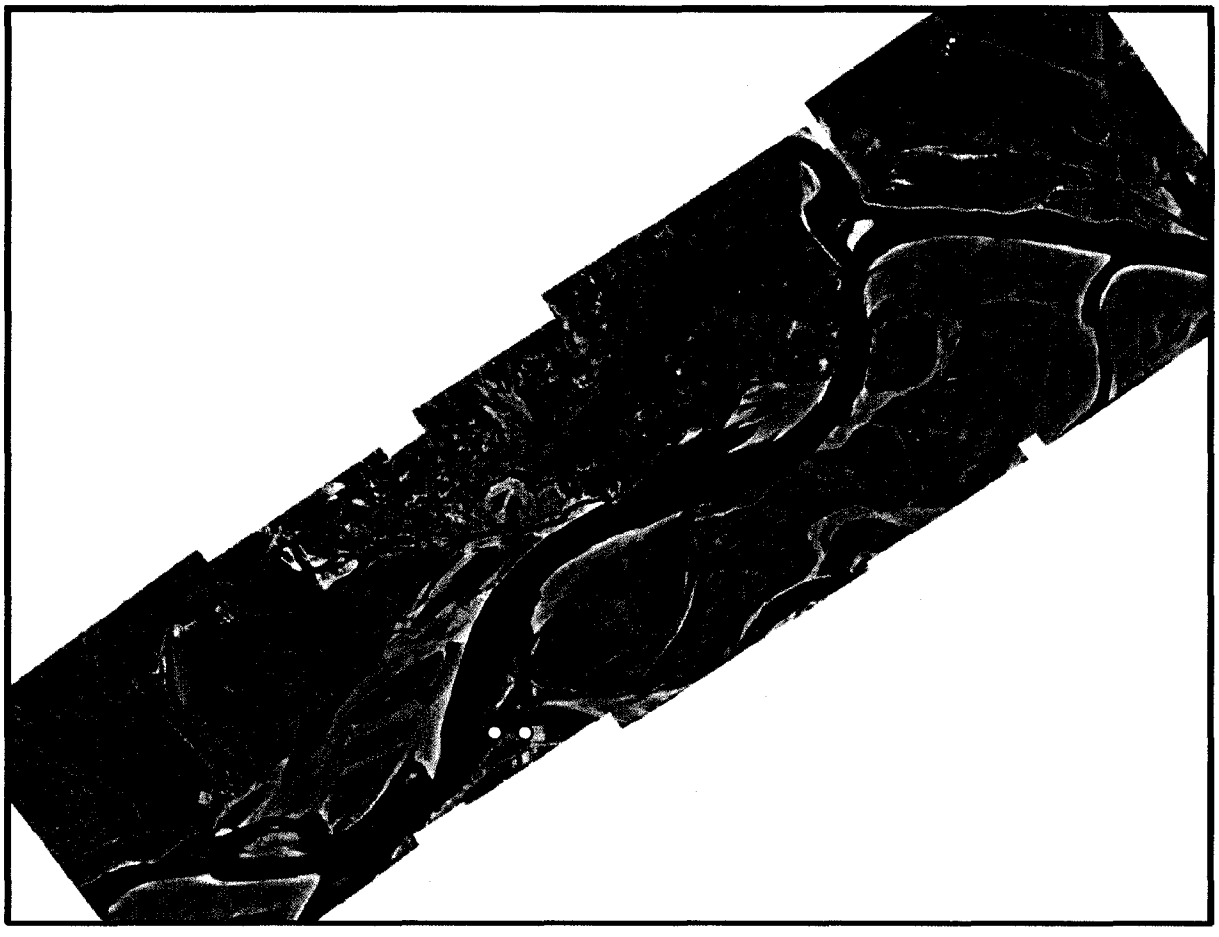

Figure 3-9 Air photo mosaic of study area, Fraser River, British Columbia (Michael Church, Department of Geography, University of British Columbia). The upstream rip-rap and the vortex locations are marked with the yellow circles.

The Figures 3-5 to 3-15 (the following pages) are the pictures of the eroding campground bank, taken from the boat on the Fraser River in sequence as the boat has moved downstream. In these pictures, the separation zone can be seen where the color of the water changes from bright to dark.

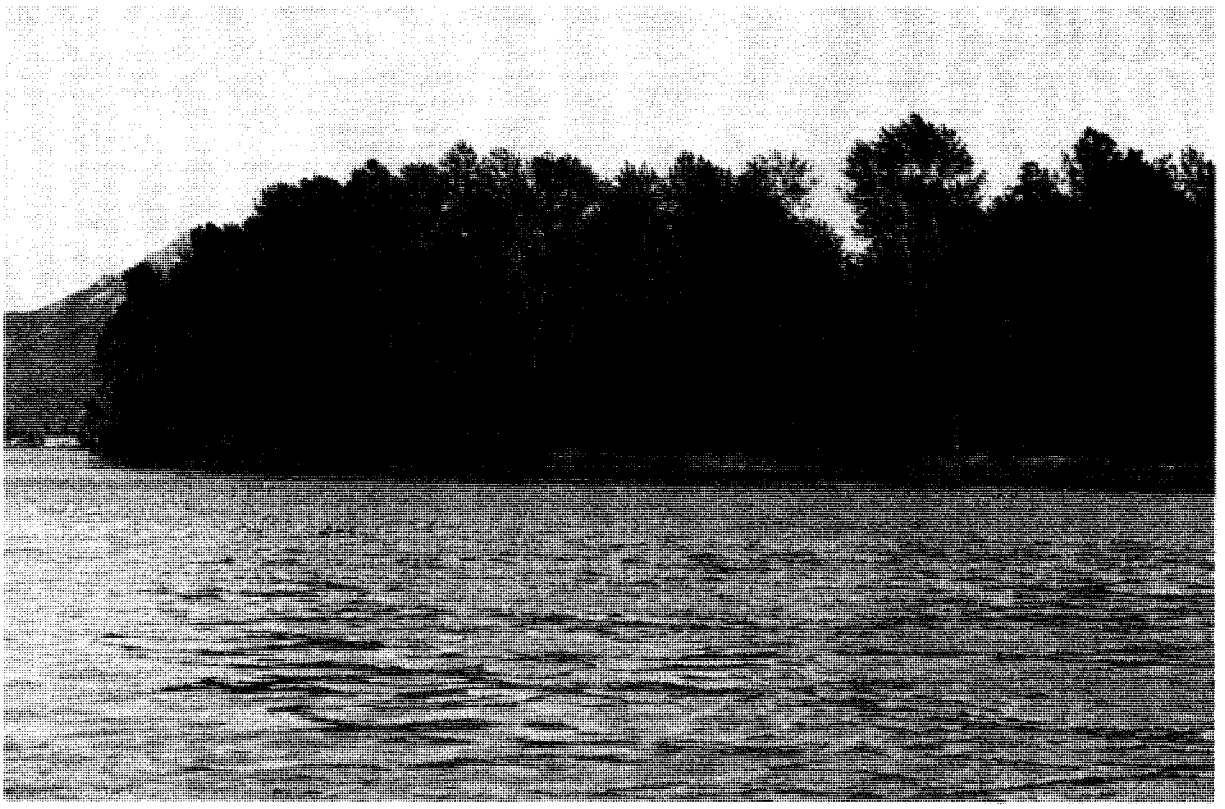

Figure 3-10 The campground bank at the confluence of the Fraser River and the Minot Side Channel in a flood season taken on the Fraser river (June 12, 2007). The Minto Side Channel is on the left side of the picture.

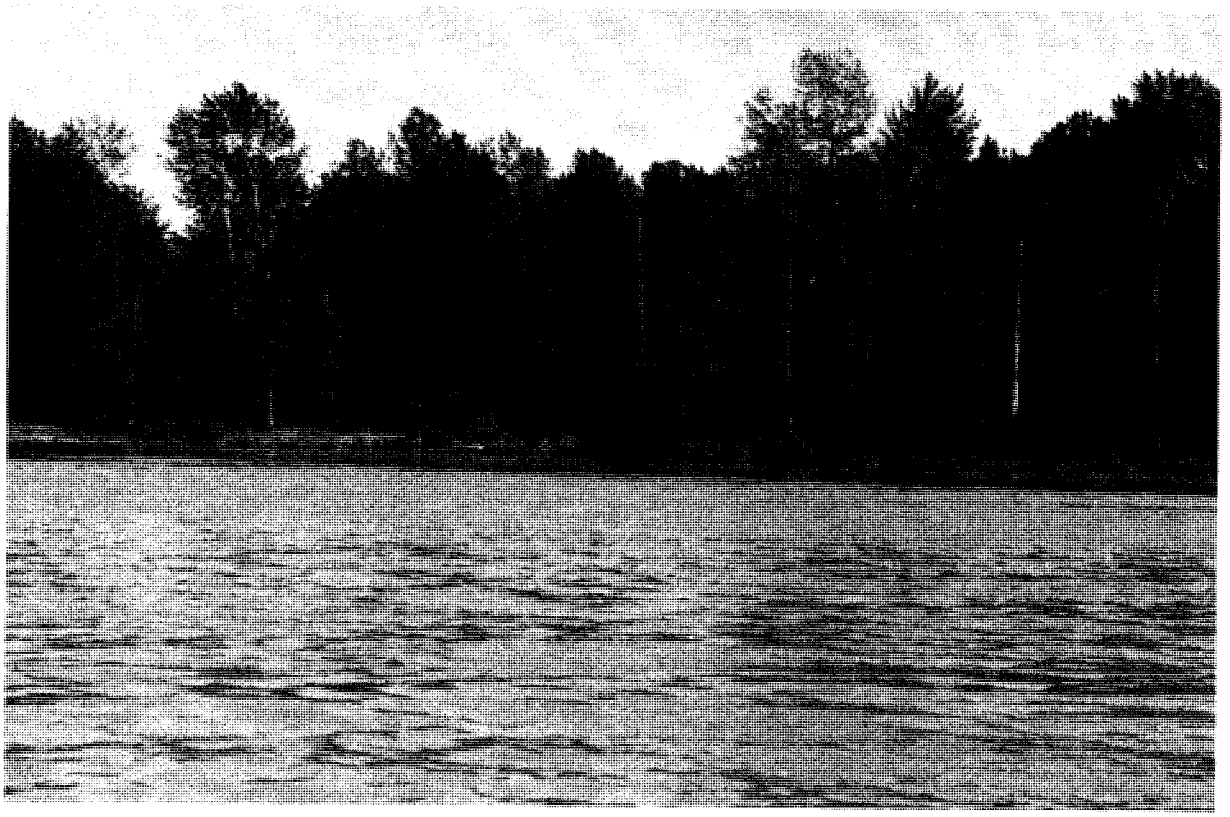

Figure 3-11 The campground bank at the confluence of the Fraser River and the Minot Side Channel in a flood season taken on the Fraser river (June 12, 2007). This picture shows downstream of the previous picture.

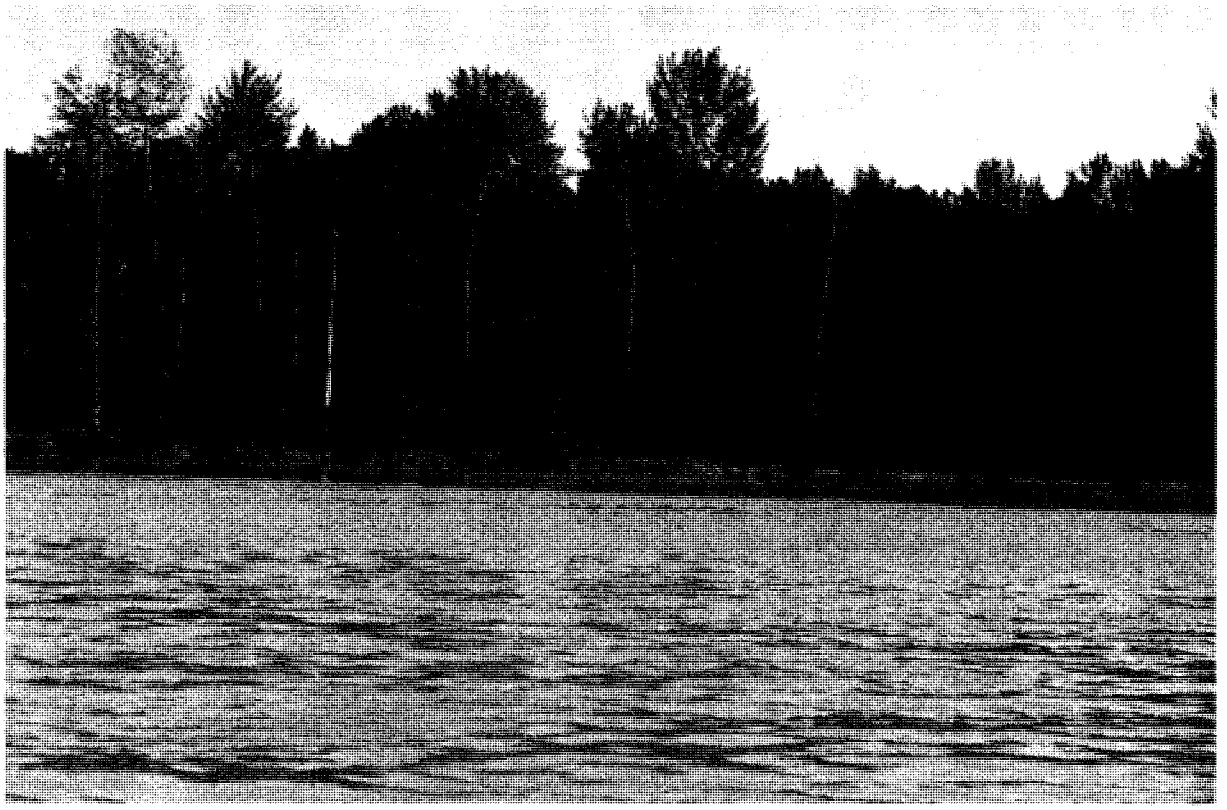

Figure 3-12 The campground bank at the confluence of the Fraser River and the Minto Side Channel in a flood season taken on the Fraser river (June 12, 2007). This picture shows downstream of the previous picture.

Figure 3-13 The campground bank at the confluence of the Fraser River and the Minto Side Channel in a flood season taken on the Fraser river (June 12, 2007). This picture shows downstream of the previous picture.

Figure 3-14 The campground bank at the confluence of the Fraser River and the Minto Side Channel in a flood season taken on the Fraser river (June 12, 2007). This picture shows downstream of the previous picture.

Figure 3-15 The campground bank at the confluence of the Fraser River and the Minto Side Channel in a flood season taken on the Fraser river (June 12, 2007). This picture shows downstream of the previous picture.

Figure 3-16 The campground bank at the confluence of the Fraser River and the Minto Side Channel in a flood season taken on the Fraser river (June 12, 2007). This picture shows downstream of the previous picture.

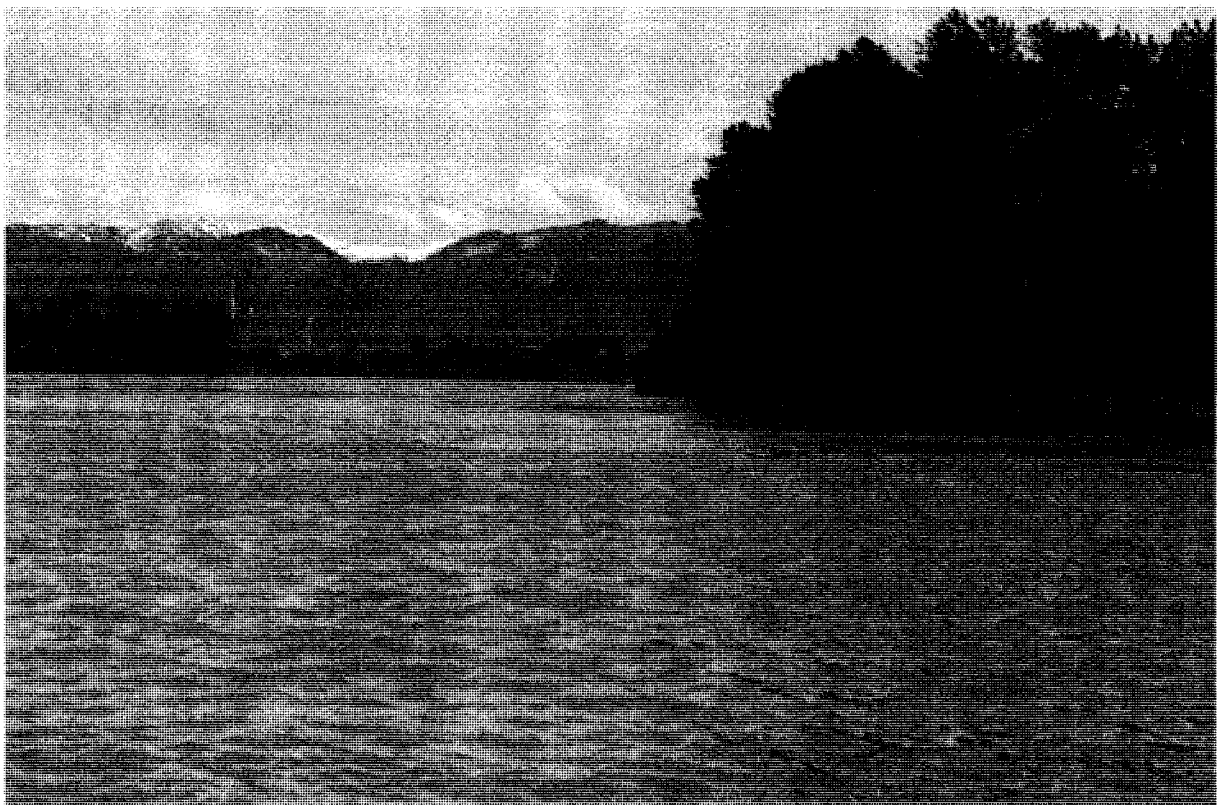

Figure 3-17 The campground bank at the confluence of the Fraser River and the Minto Side Channel in a flood season taken on the Fraser river (June 12, 2007). This picture is looking towards upstream and the confluence.

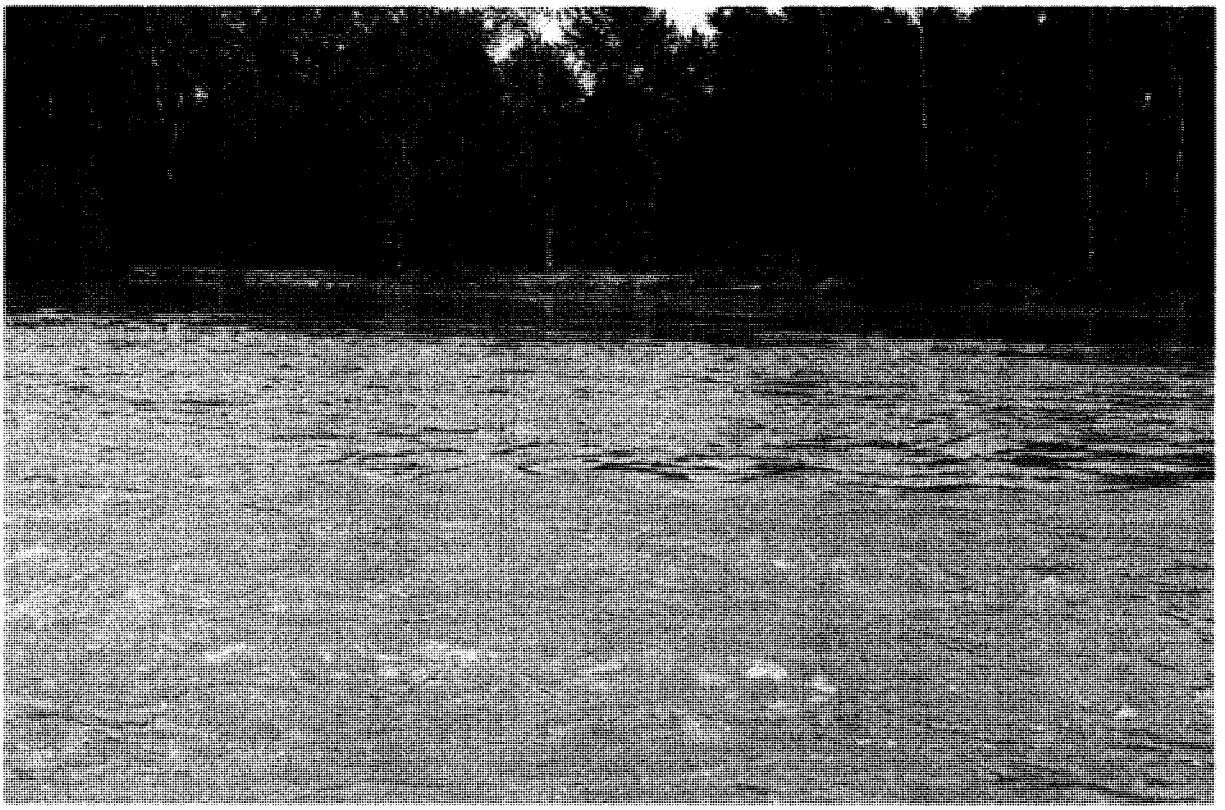

Figure 3-18 The campground bank at the confluence of the Fraser River and the Minto Side Channel in a flood season taken on the Fraser river (June 12, 2007). Boils due to the turbulence and vortices are visible in this picture.

3.2 Data Collection

The data were collected on June 24-25, 2006 in the spring freshet, when the average discharge was 6000 m^3/s (based on the Fraser River at Mission gauge), and the bed was moving near the threshold of motion). The survey was made on an aluminium boat (Figure 3-19 shows the boat course), using an RD-Instruments four beam 1200 kHz broadband Rio Grande ADCP and a NovAtel RTK-GPS. The mount of the ADCP and GPS was also made of aluminium, since the use of ferrous material would affect the ADCP compass and would

result in erroneous data.

The water velocity and depth were measured by the ADCP, and location and satellite time by the GPS. The ADCP and GPS were connected to a laptop. The ADCP and the GPS were run using the WinRiver program and their data were saved in three file formats-w files, r files, and n files, which are configuration files, navigation files, and raw data files, respectively. These files are created simultaneously. The n files contain the GPS data and are in ASCII format. The r files are in binary format and contain the ADCP data as well as the GPS time and position data every 0.6 s, i.e. at every moment that the ADCP sent a ping. In n files, $RDENS lines show the moment that the GPS data were sent from the n file to the respective r file (appendices A.4 & A.5). In every run of the survey, one set of the w, r, and n files were created. In June 24-25, 2006 six runs of survey were made to cover the reach, and therefore, six set of files were created (appendices A.1 A.2 A.3).

The ADCP measurement of the flow velocity and depth were made using 25 cm long bins in single ping ensembles which were emitted almost every 0.6 s. The ADCP transducers were located 36 cm below the water surface and the blanking depth was 25 cm below ADCP transducers. Thus, data were collected starting at 61 cm below the water surface down to the bottom of the river. The bottom 6% of each profile was excluded to avoid side-lobe errors (Simpson 2001). The GPS measurement of the location and satellite time was made every 0.1 s.

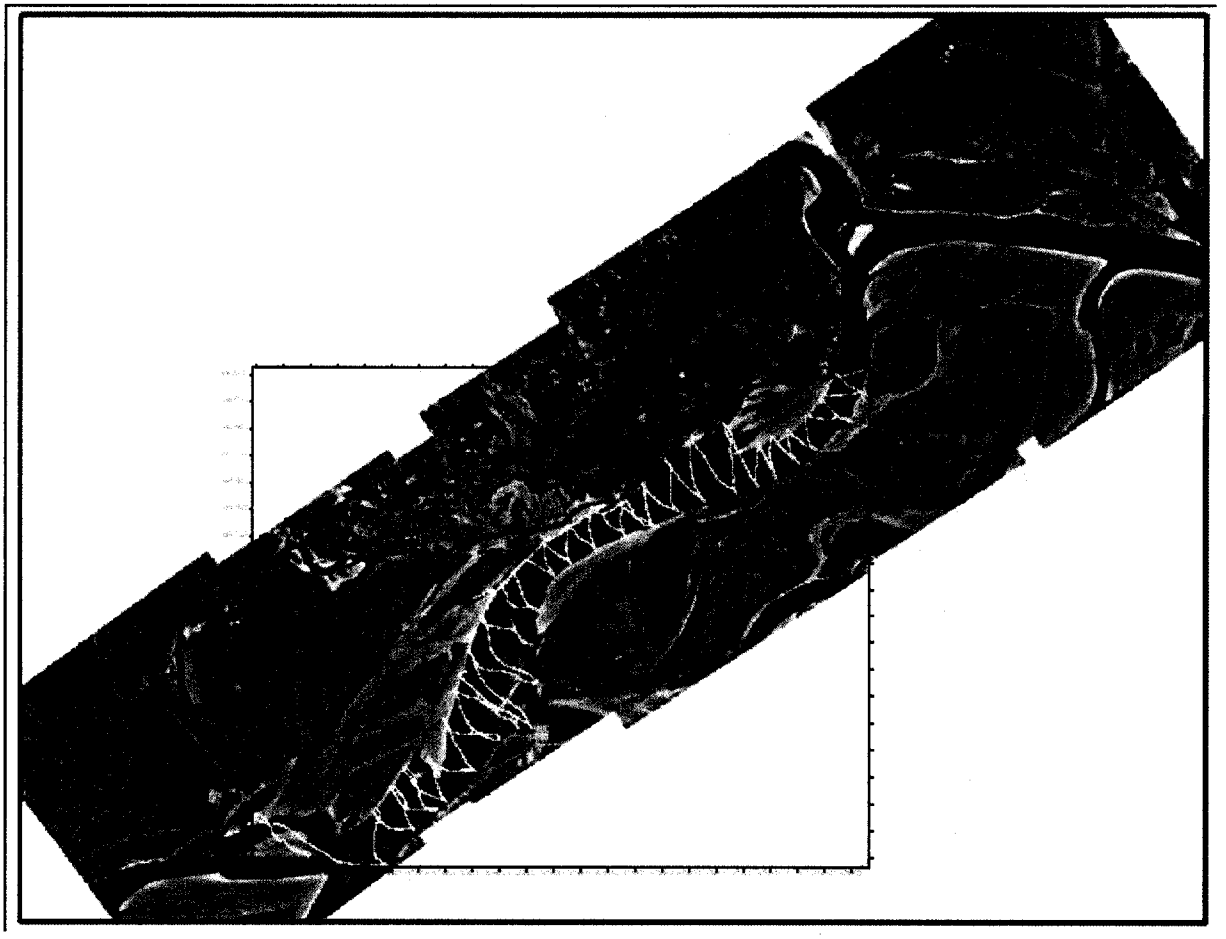

Figure 3-19 The boat course where survey and data collection were made (Rennie and Church, 2007).

3.3 Data Analysis

3.3.1 Data Extraction and Correction

Six survey runs were made in June 24-25, 2006 and therefore six pairs of r and n files were created. The fifth set of files had too many bad data and was eliminated from the analysis. So, the data from the first to fourth run and from the sixth run were used.

Using Matlab, the data in the r files were extracted. The millimetre precision elevation and time of every GGA line right before RDENS lines in the n files were extracted as well. The vertical boat velocity for every ensemble was calculated by dividing the elevation differential of that ensemble and the previous one by its respective time differential. The reason that n files were used for this purpose was that r files store the elevations in decimeter precision and not millimeter precision which was needed for accurate calculation of boat vertical velocity.

Using Matlab, the utme, utmn, water surface elevation, and the boundary elevation for every ensemble were extracted from the r files. The utme, utmn, elevation, velocity-east, velocity-north, and velocity-vertical for every bin in each ensemble were also extracted form the r files. The boat vertical velocity of each ensemble, calculated from the n files, was subtracted from the vertical velocities of all the bins in the same ensemble to remove the error caused by the boat and the ADCP vertical movement.

The other data correction that had to be made was to fix error due to GPS-satellite miscommunication. The Harrison Knob and the Harrison Hill, downstream of the Harrison confluence, had shaded the GPS from proper satellite communication. As a result, the elevations in part of the reach beside the Harrison Knob and the Harrison Hill were erroneous and much higher than expected. For example, the water surface elevation in that area was measured to be up to 25 m, whereas the elevation of the adjacent points indicated that an elevation of 8.25 m would be the normal water surface elevation of that area. This problem was fixed by rewriting the Matlab code (extractfortecplot.m) such that GPS water surface elevation of that area (575373-575598 utme, 5451528-5451728 utmn) was fixed at 8.25 m. The river boundary elevation and the elevation of the bins in every ensemble were also modified to correspond to the modification made in the water surface elevation in the same ensemble.

In Matlab, another change in the data set had to be made so that in the next stages they could be used in Tecplot. The problem related to utilizing Tecplot for interpolating velocity data to

a uniform 3D finite volume grid. Regardless of whether or not the input data would represent a slanting water surface, Tecplot would only provide the option of entering a horizontal water surface elevation while creating the prism (finite volume) grid for velocity interpolation. Since the water surface elevation at the upstream of the reach was 2 m higher than the elevation at the downstream, this would cause a problem in Tecplot, i.e. this 2 m of elevation difference would be neglected in the model created by Tecplot. To solve this problem, using Matlab, the elevation of every point was transferred so that the water surface elevation would be a horizontal surface. This was done by multiplying the average slope of the surface by the distance of every ensemble from the downstream of the reach, and subtracting the resulting number from the elevation of every point of the water surface, river boundary, and velocity bins in the same ensemble. For this research project, this transfer will not cause any problems, since the water surface elevation, boundary, and velocity vectors stay in the same position relative to each other.

Specific filenames used in the analysis will be described herein, in order to clarify the content of the appendices. For this research, three files in ASCII format are the output of Matlab preprocessing: the file tecplotwatersurfexportascii which contains utme, utmn, and water surface elevation for each ensemble, the file tecplotboundaryexportascii which contains utme, utmn, and river boundary elevation in each ensemble, and the file tecplotmatrixexportasciiwitherror which contains utme, utmn, elevation, velocity-e, velocity-n, velocity-ver, and error velocity for every bin in each ensemble (appendix D).

3.3.2 Water Surface and Boundary Elevation Interpolation

Using Surfer, the water surface elevations and the boundary elevations were interpolated to create uniform grids of the water surface elevation and the boundary elevation.

To do so, the file tecplotwatersurfexportascii, which includes the water surface elevation data, was loaded into Surfer. Then the variogram which best fitted the experimental data was

modelled (appendix C). Then the data were interpolated into a 25 m x 25 m gird, using kriging method. Velocities from individual pings are unreliable, thus averaging is required. Kriging employs a model of the spatial variance structure of the field (the variogram) when utlizing neighbouring measured values to estimate values at grid points, but generally gives more weight to nearby values and less weight to the values far away. A grid of 25 m x 25 m was a good size for interpolation, because it was large compared to the measurement spacing along a transect of approximately 1 m. This ensured that a sufficiently large number of pings was available for grid points along a transect. Error sources, and subsequent errors in the estimated spatial distributions, will be reviewed in the discussion.

Points of the generated grid that were outside of the reach were blanked. The resulting file was exported into an Excel spreadsheet. Then the file tecplotboundaryexportascii, which includes the river boundary elevation data, was loaded into Surfer and interpolated into the same 25 m x 25 m grid by the same process. The resulting file was also exported into an Excel spreadsheet and then combined with that of the water surface elevation into one file, named boundarysurface, to be used in Tecplot to create a 3D image of the reach boundary and water surface elevation (Figure 3-20).

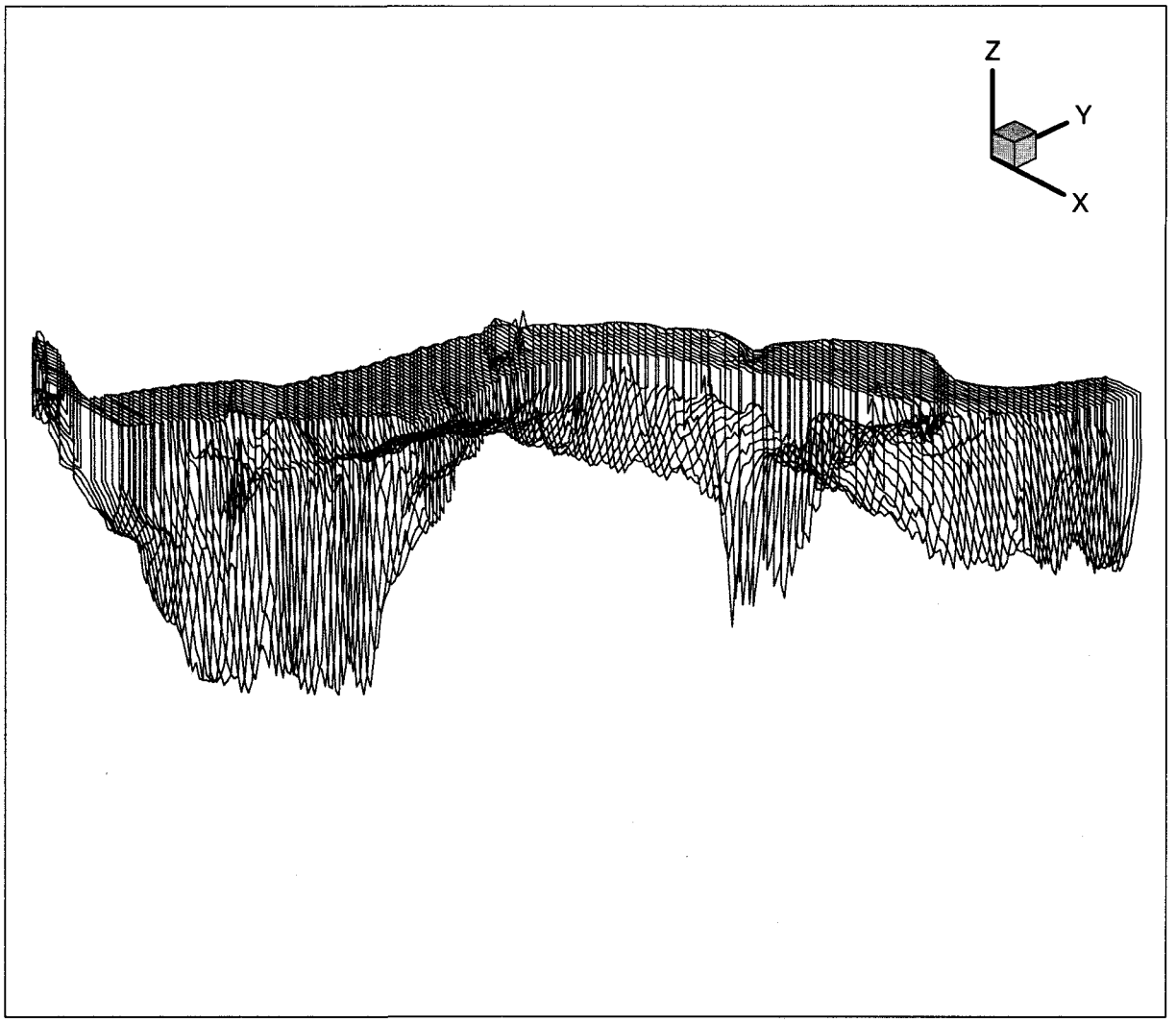

Figure 3-20 3D image of the reach boundary and water surface elevation. The boundary and water surface elevation are interpolated in Surfer and the interpolated data are used to create the image in Tecplot. The vertical scale is 100 times the horizontal scale. Flow direction is from right to left.

3.3.3 Velocity Interpolation

The Tecplot program was used to create the three-dimensional image of the reach boundary and to interpolate the velocity data into a volume gird. To do so, first the file boundarysurface was loaded into Tecplot. Then, based on the 25 m x 25 m horizontal grid of

the file boundarysurface, a volume prism grid with 10 vertical layers was created within the river reach volume. Then the file tecplotmatrixexportasciiwitherror, which contains the velocity and error velocity data, was loaded into Tecplot (Figure 3-21 and 3-22) and interpolated into the prism gird using kriging. Vorticities were also calculated in Tecplot. The interpolated velocities, error velocities, and vorticities are presented in the next chapter (Figures 4-1 to 4-21).

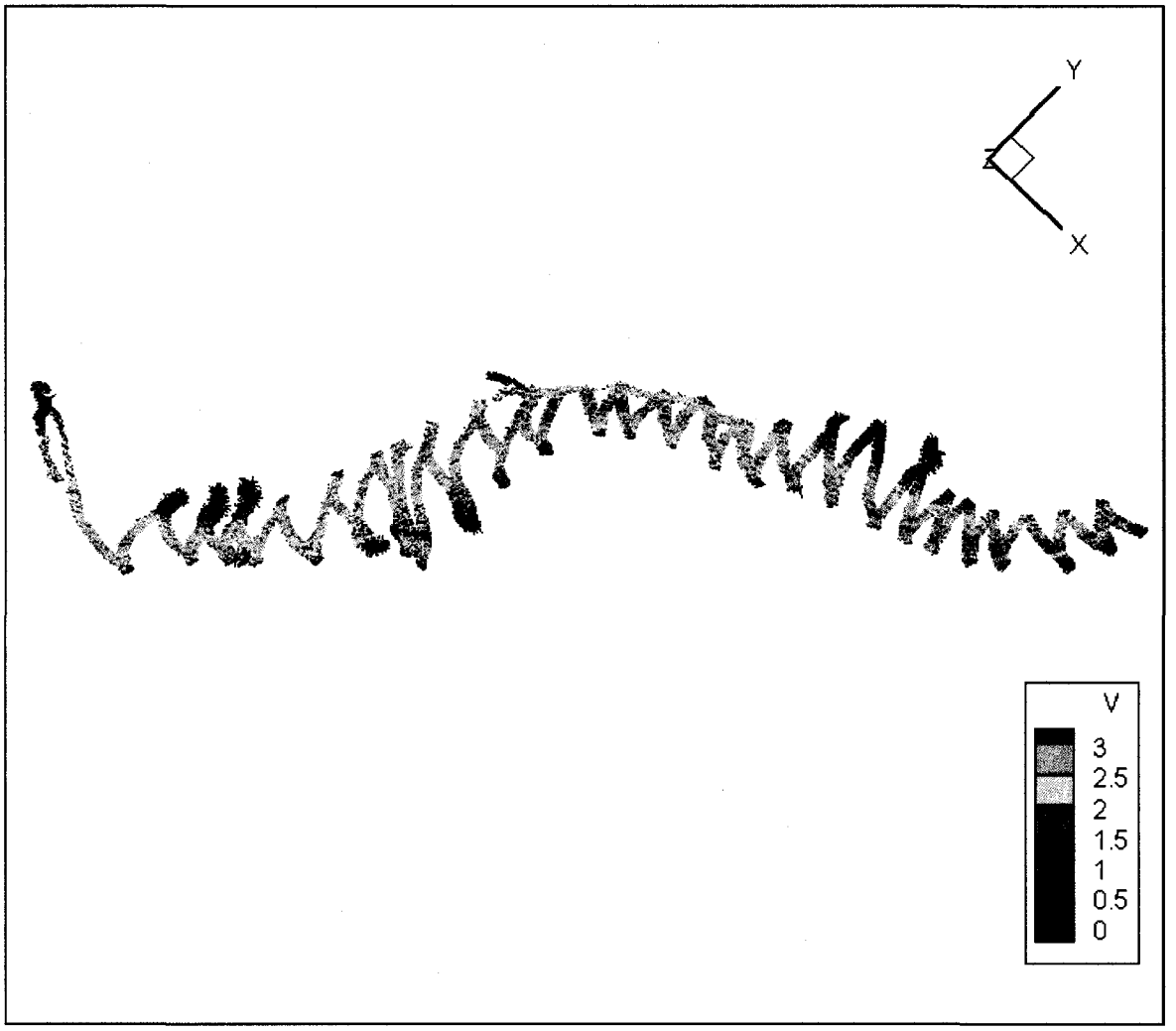

Figure 3-21 Top view color scaled image of raw velocity data loaded into Tecplot. Flow is from right to left. Color scale is based on velocity magnitude (m/s). Reach length is 5.5 km. The plot reveals that high velocity

follows the thalweg.

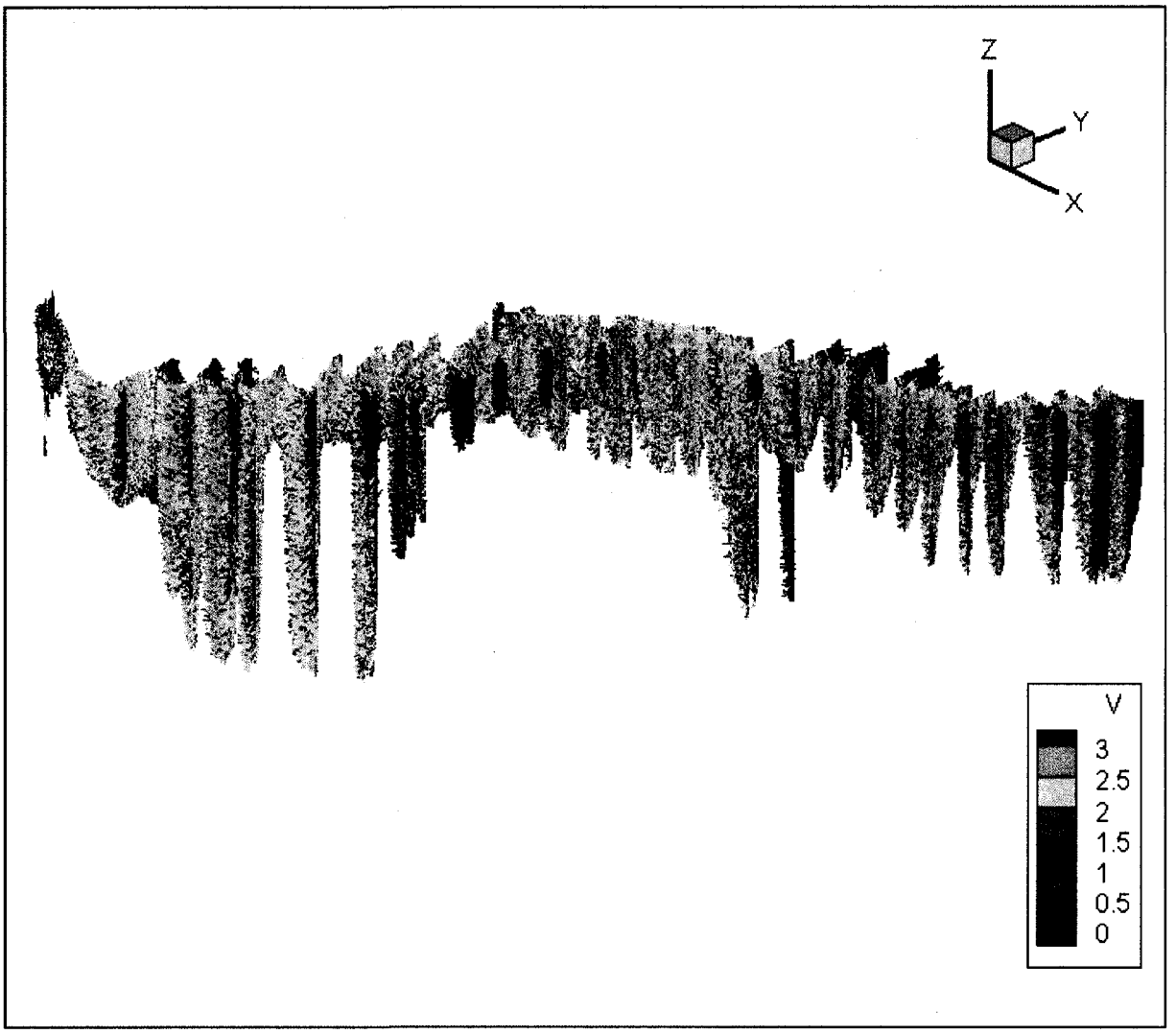

Figure 3-22 3D color scaled image of row velocity vectors loaded into Tecplot. Flow is from right to left. The vertical scale is 100 times the horizontal scale. Color scale is based on velocity magnitude (m/s).

4 RESULTS

The three-dimensional images of flow velocity, error velocity, and vorticity are prepared by intensive surveying of the Fraser River on a boat using a combination of ADCP and GPS, and by analyzing the data by Matlab, Surfer, and Tecplot programs. The error caused by vertical boat motion is calculated using the GPS position and time data, and is subtracted and removed from the velocity data collected by ADCP using Matlab. The images are presented in Figures 4-1 to 4-24. These figures are color scaled based on the magnitude of the respective velocity or vorticity component, however, Tecplot plots the direction of the vectors in all plots just based on the direction of the velocity vectors and does not show the real direction of the velocity or vorticity components. Furthermore, the colour range for each plot has been selected to exclude outliers, thus the colour ranges do not represent the full data ranges.

The coherence of these images reveals that this method, i.e. surveying river on a boat using a combination of ADCP and RTK-GPS, is an efficient method for preparing 3D images of flow velocity field and vorticity in such a large scale. In the vertical direction, velocity increased towards the water surface. Lower velocities were observed near channel boundaries. High velocities occur in the areas of flow convergence where the thalweg and pools are formed, and low velocities occur in the areas of flow divergence where point bars have taken shape (Figure 4-1).

The low values of the calculated error velocities also indicate on the coherence of the method. Figure 4-23 reveals that the value of error velocity for the most of the measurement in the reach is between -0.2 m/s and 0.2 m/s. Since error velocity is scaled based on the standard deviation of the horizontal velocity, the range of -0.2 m/s to 0.2 m/s for the error velocity shows that the measured error is relatively low, in comparison to the thalweg

horizontal velocity of between 2 m/s and 3 m/s (Figures 3-21, 3-22, 4-1). Figure 4-24 is color scaled to better show the error velocity for the values between -0.2 m/s and 0.2 m/s. It reveals that error velocity, as expected, is higher in the deeper areas. The reason is that the distance between the beams increases with the increase of the depth, and therefore, a less homogenous measurement is made by the four beams.

However, in the confluence of the Fraser River with the Minto Side Channel, beside the campground, a deep pool exists and the bank adjacent to it is being eroded and retreating rapidly. This is in spite of the fact that this point is located upstream of the thalweg, in a location of relatively low flow velocity and shear stress.

The three-dimensional images of flow velocity field and vorticity (Figures 4-2 to 4-22) reveal that, in spite of relatively low horizontal velocity, the vertical velocity and vorticity are relatively high in this area. The inflow of the Minto Side Channel to the Fraser River and its combination with the armouring of the bank upstream of the eroding bank has led to high turbulence and high vertical velocity which appear to be the cause of high erosion in this area.

In the eroding bank, the horizontal velocity in the x direction is relatively low compared to the maximum x velocity of the reach. In the vicinity of the eroding bank, the water velocity in x direction is between zero and 0.55 m/s, i.e. less than 25% of the maximum water velocity in x direction, which is 2.2 m/s (Figures 4-2 to 4-4). The horizontal velocity in y direction is also relatively low compared to the maximum y velocity of the reach. In the vicinity of the eroding bank, the water velocity in y direction is between zero and 0.55 m/s, i.e. less than 25% of the maximum water velocity in y direction, which is 2.2 m/s (Figures 4-5 to 4-7). However, the vertical velocity in the z direction is relatively high compared to the maximum z velocity of the reach. In the vicinity of the eroding bank, the water velocity in z direction is equal to the maximum water velocity, which is 0.10 m/s. The other significant fact about the vertical component of the flow is that the velocity is both upwards and downwards, which is

an indicator of high vertical turbulence (Figures 4-8 to 4-10). So, in spite of relatively low horizontal velocity, high vertical velocity and turbulence appear to cause bank erosion at the campground.

In the eroding bank, the vorticities about x and y and z axes are relatively high. This confirms the existence of high coherent turbulence in this zone and its role in the bank erosion. Figures 4-11 to 4-13 reveal that, in the eroding bank, vorticity about the x axis is relatively high and, in a considerable part of the study zone, close to the maximum value of 0.3 s^{-1}. Figures 4-14 to 4-16 reveal the existence of high vorticity and turbulence about the y axis. They reveal that, in a considerable part of the study zone, the y vorticity is close the maximum value of 0.3 s^{-1}. Figures 4-17 to 4-19 reveal the same patterns for the vorticity about the z axis and that, in a major part of the study zone, it is close to its maximum value of 0.02 s^{-1}. These results reveal that flow separation and high turbulence can be the cause of the erosion of the campground bank. The flow separation can be seen in the pictures of the study site presented in the Figures 3-10 to 3-18. The flow separation can be seen as the change in the color and smoothness of water surface in these pictures. Figure 3-18 also shows boils in the water surface, produced by turbulence bursting in the flow.

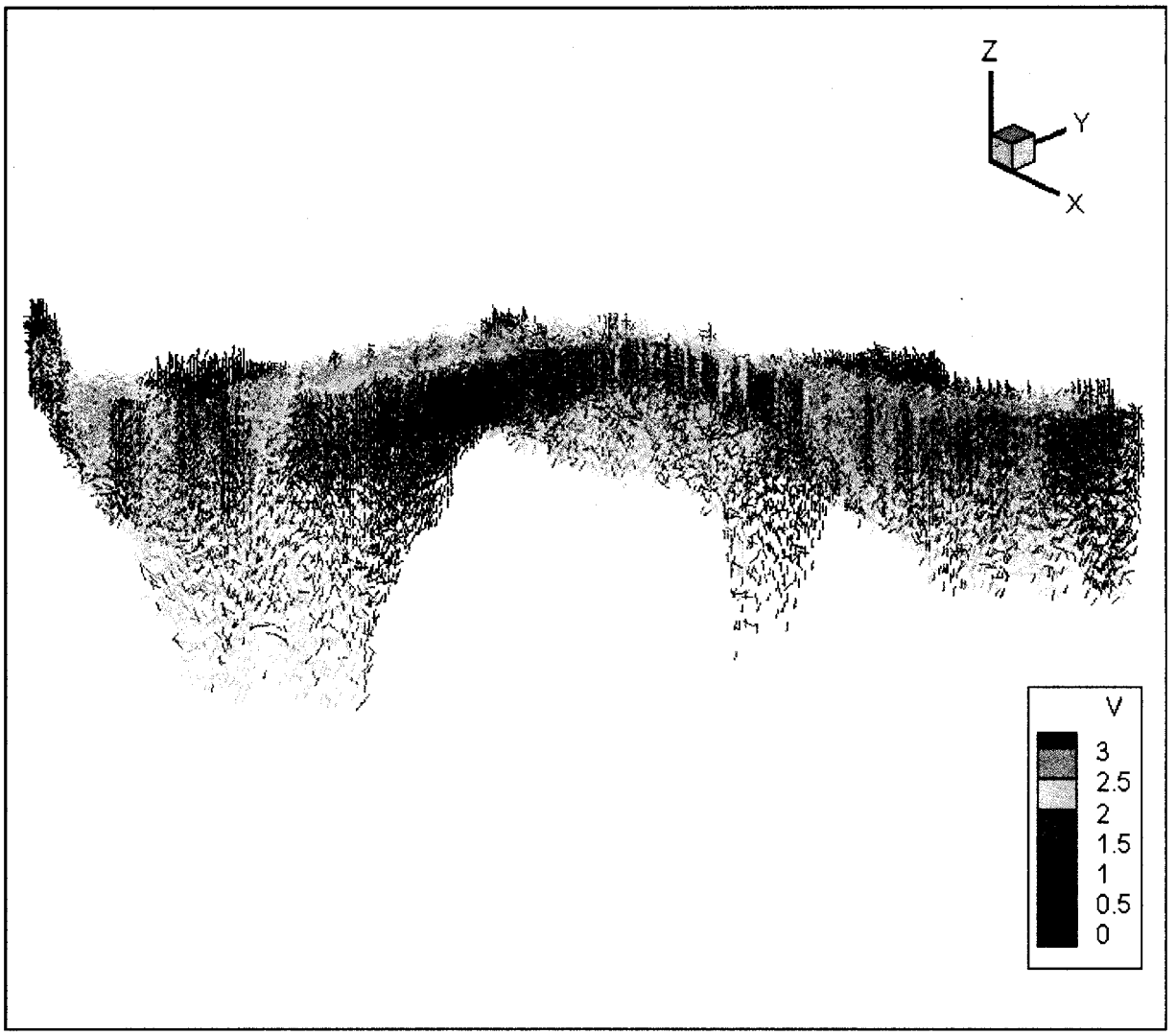

Figure 4-1 3D color scaled image of interpolated velocity (m/s). Flow is from right to left. The vertical scale is 100 times the horizontal scale.

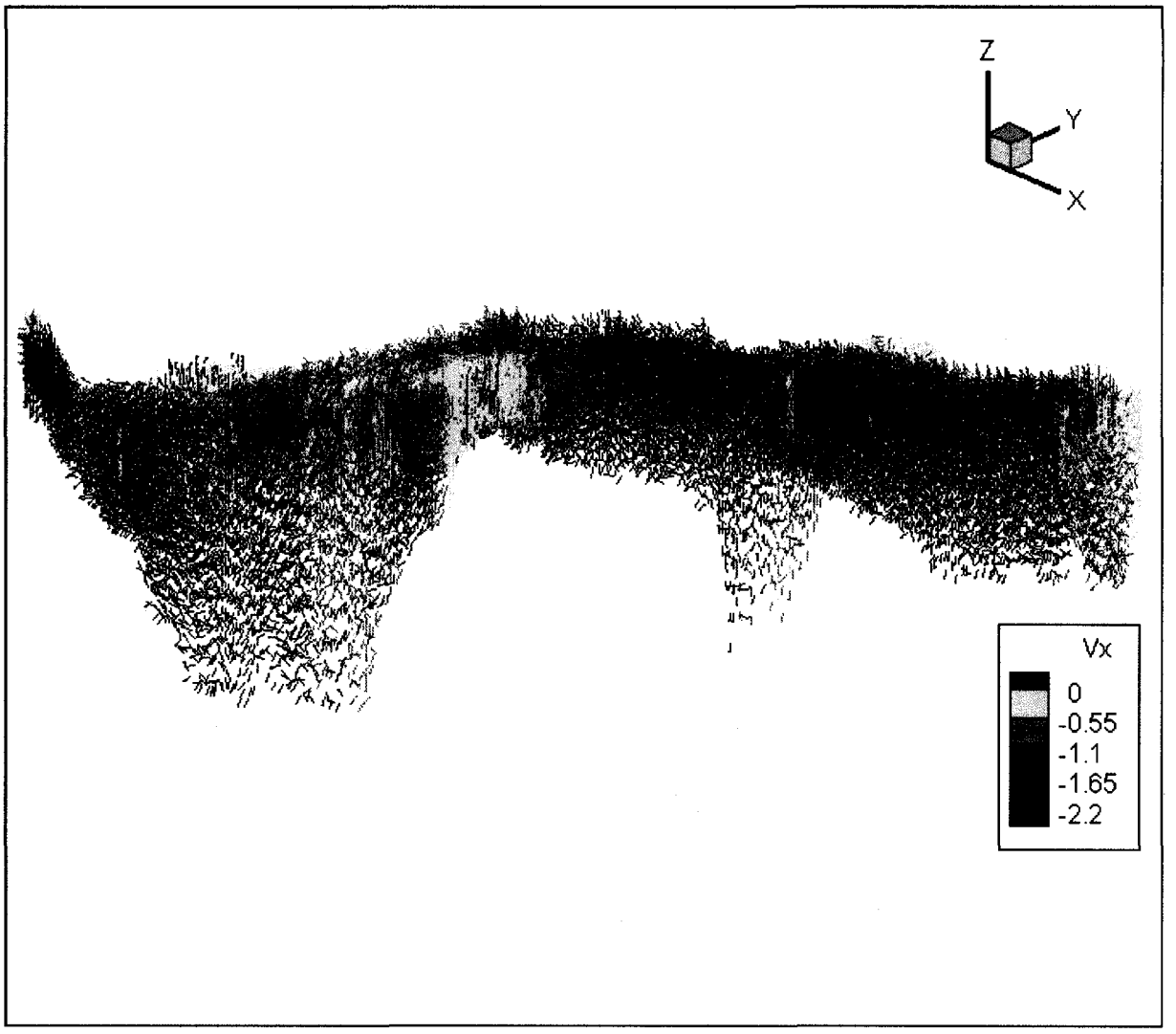

Figure 4-2 3D color scaled image of interpolated velocity component in x direction (m/s). Flow is from right to left. The vertical scale is 100 times the horizontal scale. The eroding bank is marked with the ellipse.

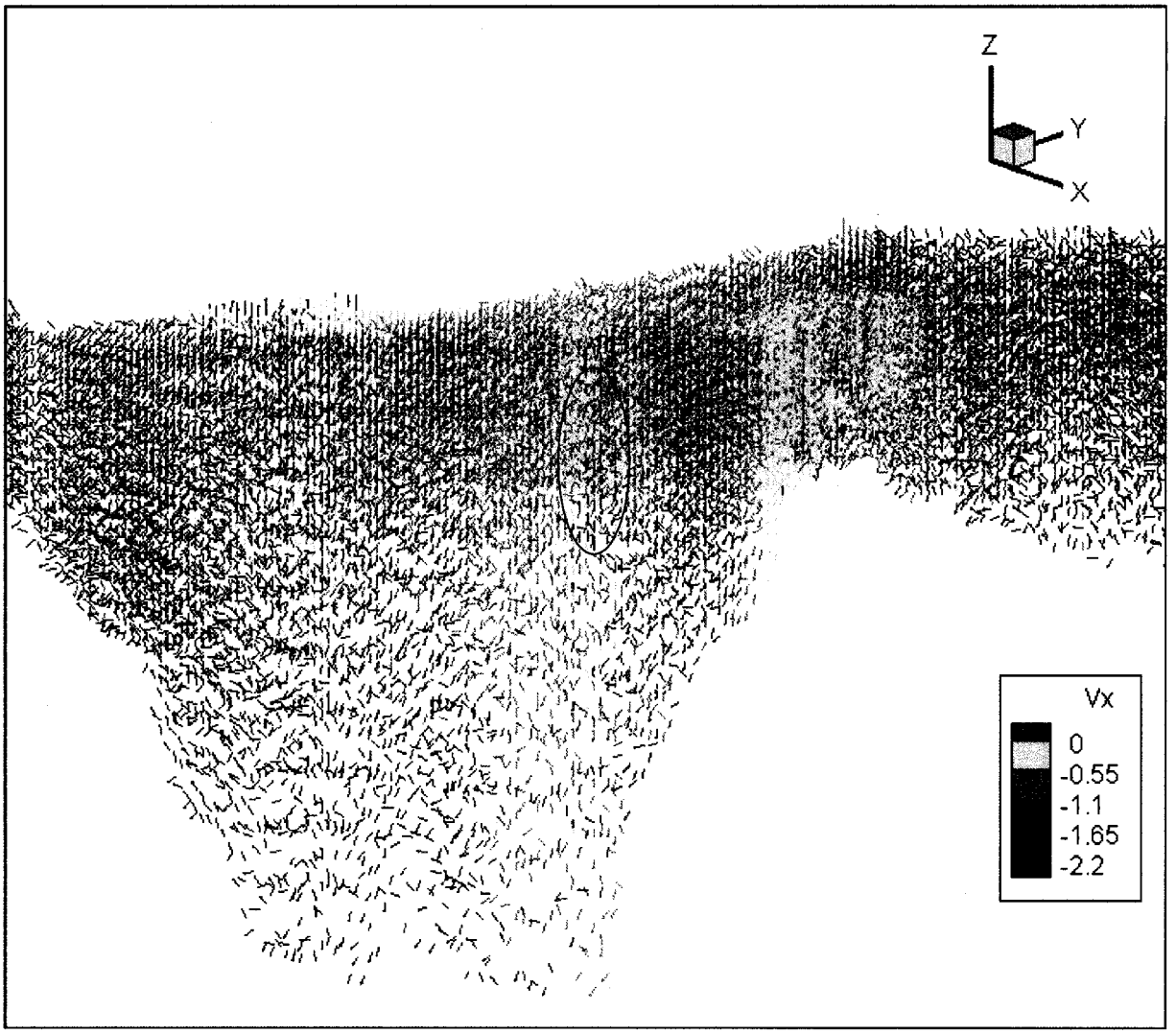

Figure 4-3 Zoomed in 3D color scaled image of interpolated velocity component in x direction (m/s). Flow is from right to left. The vertical scale is 100 times the horizontal scale. The eroding bank is marked with the ellipse.

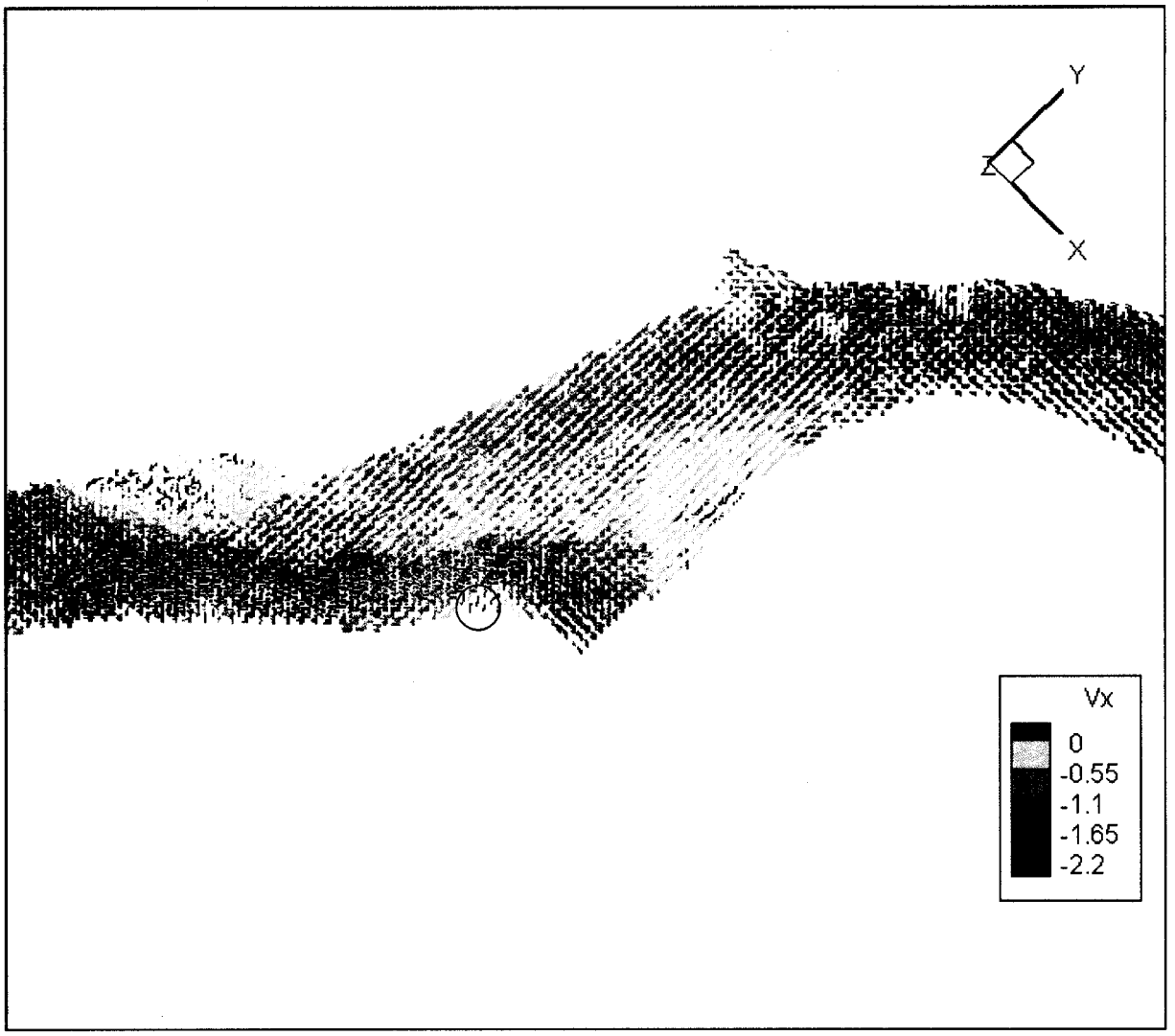

Figure 4-4 Zoomed in color scaled top view of interpolated velocity component in x direction (m/s). Flow is from right to left. The vertical scale is 100 times the horizontal scale. The eroding bank is marked with the circle.

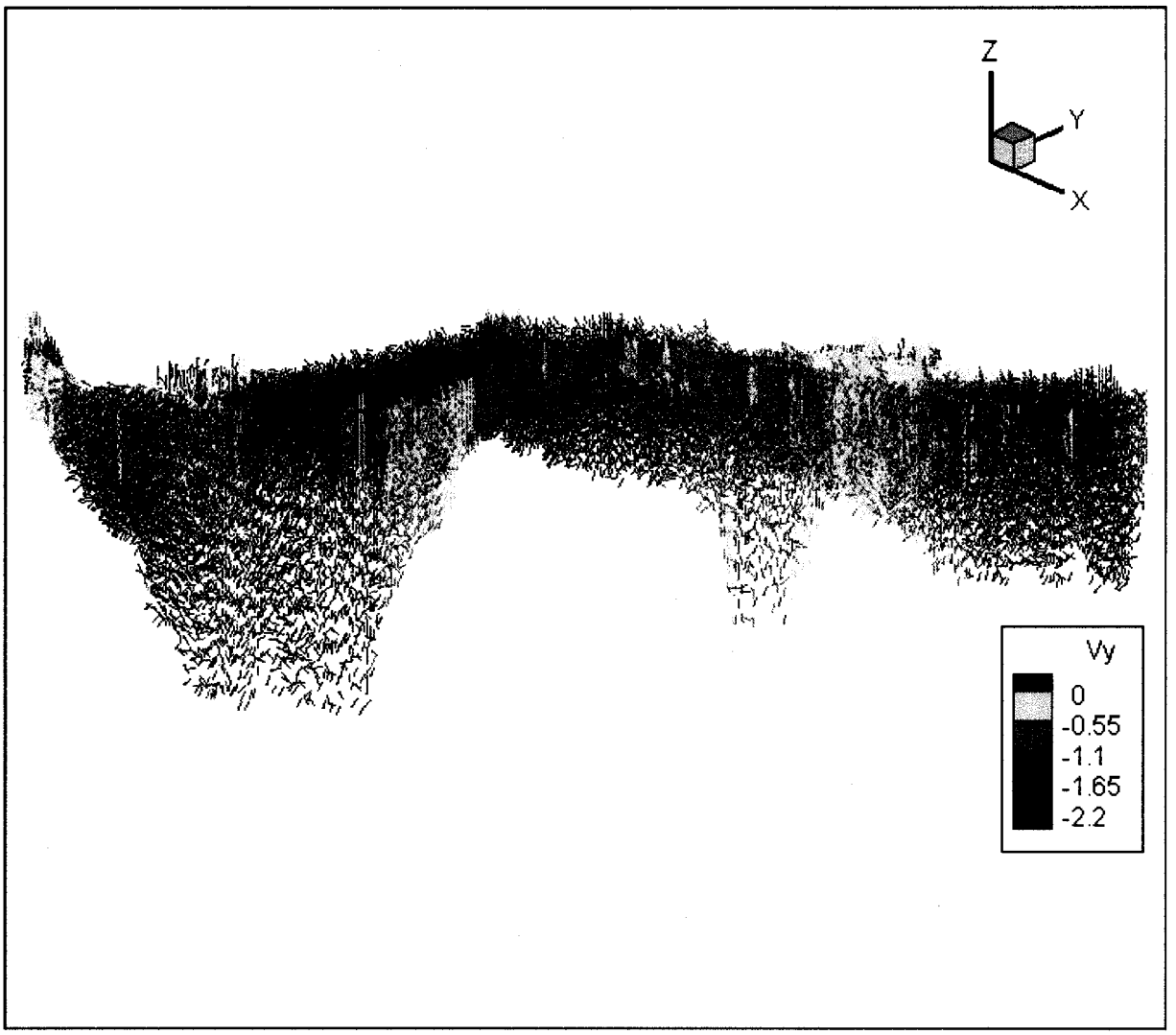

Figure 4-5 3D color scaled image of interpolated velocity component in y direction (m/s). Flow is from right to left. The vertical scale is 100 times the horizontal scale. The eroding bank is marked with the circle.

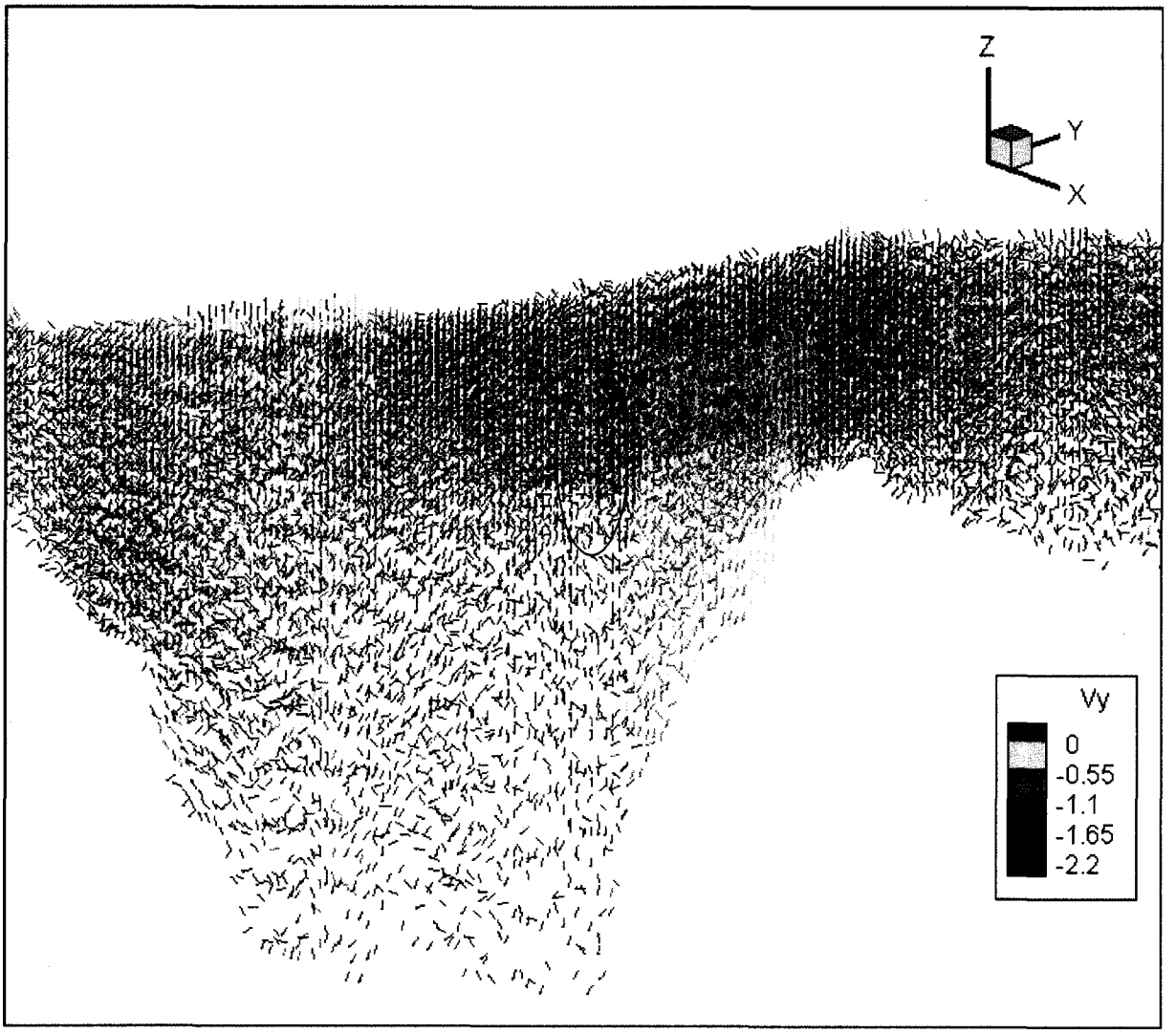

Figure 4-6 Zoomed in 3D color scaled image of interpolated velocity component in y direction (m/s). Flow is from right to left. The vertical scale is 100 times the horizontal scale. The eroding bank is marked with the ellipse.

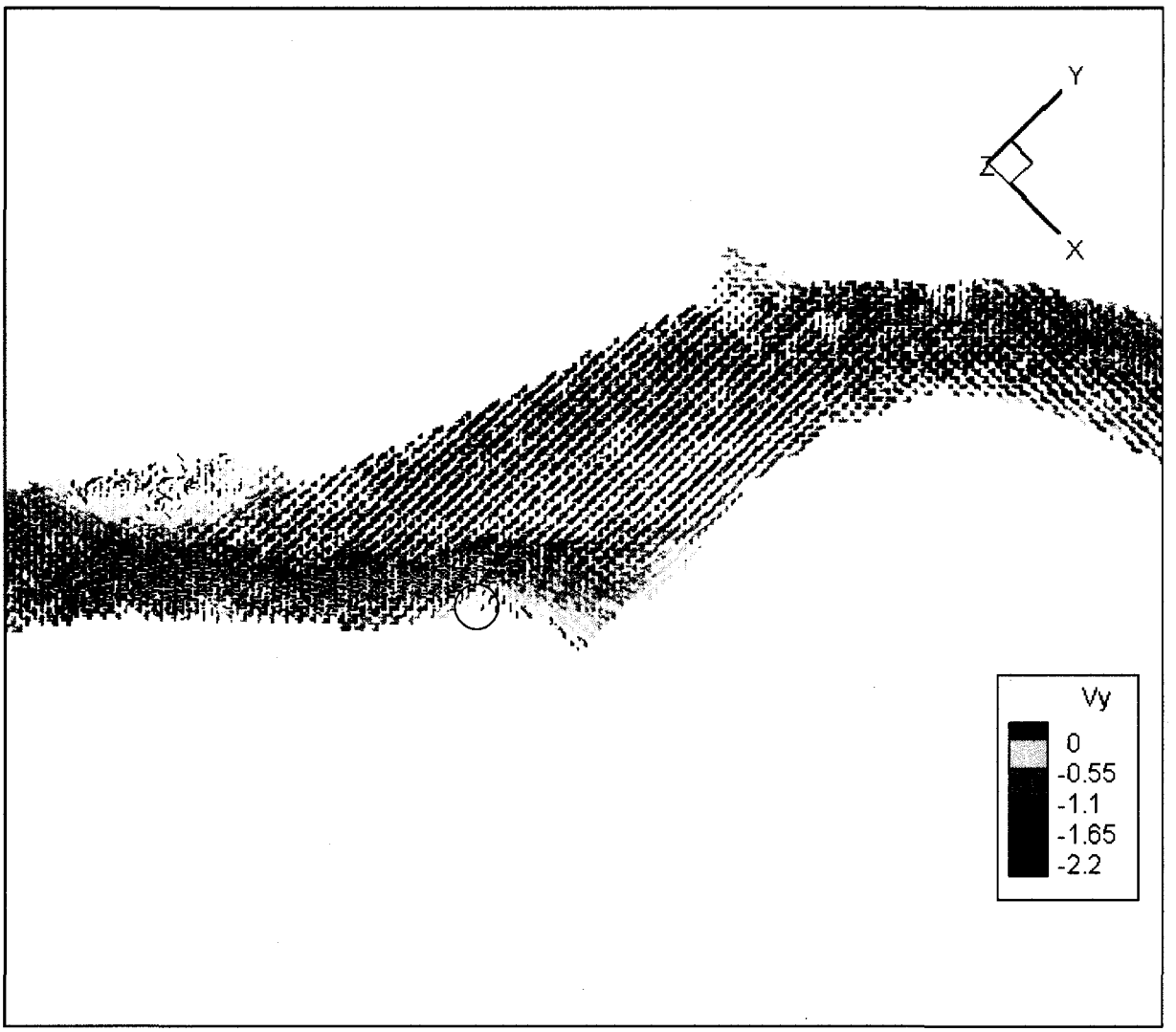

Figure 4-7 Zoomed in color scaled top view of interpolated velocity component in y direction (m/s). Flow is from right to left. The vertical scale is 100 times the horizontal scale. The eroding bank is marked with the circle.

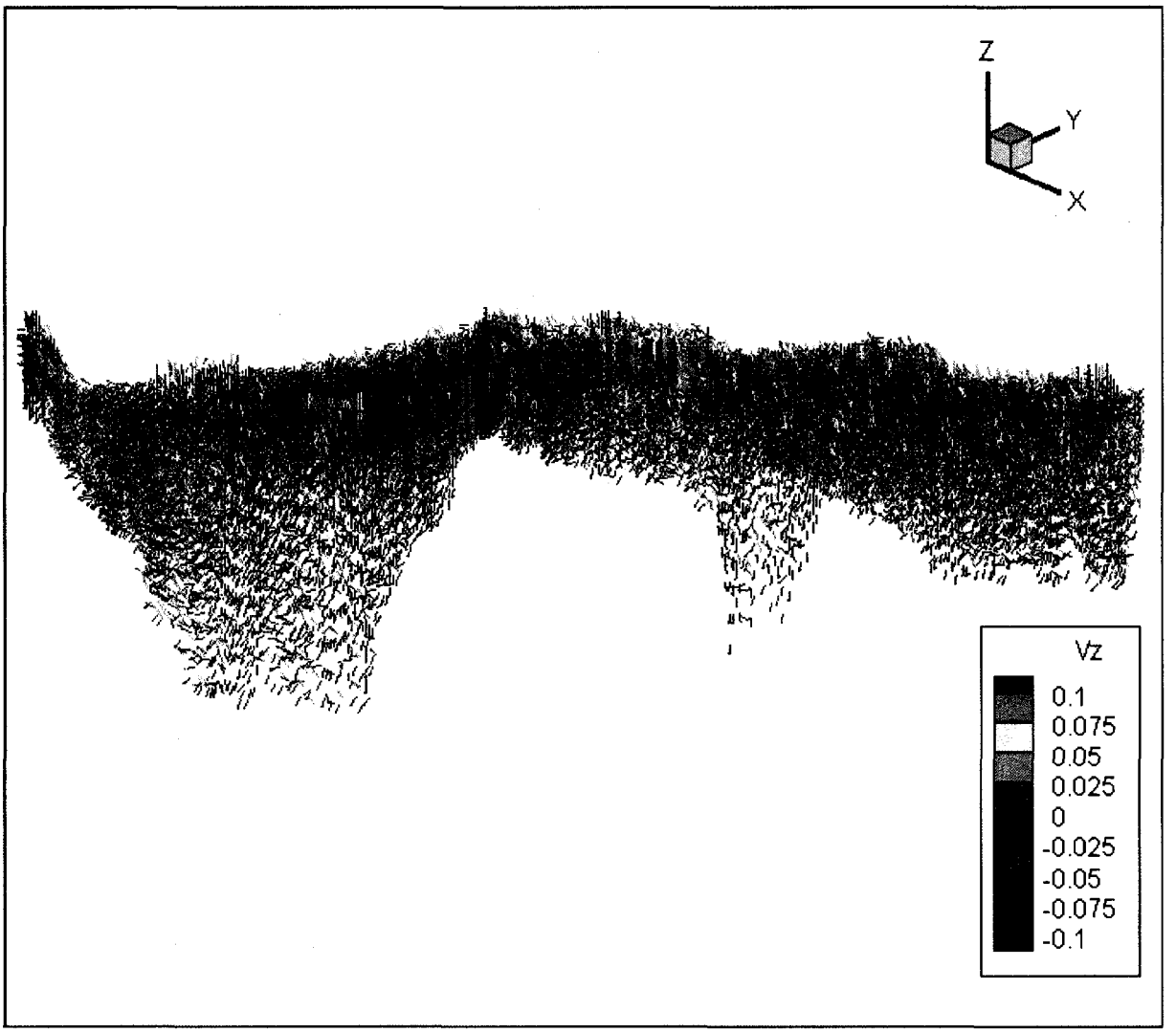

Figure 4-8 3D color scaled image of interpolated velocity component in z direction (m/s). Flow is from right to left. The vertical scale is 100 times the horizontal scale. The eroding bank is marked with the ellipse.

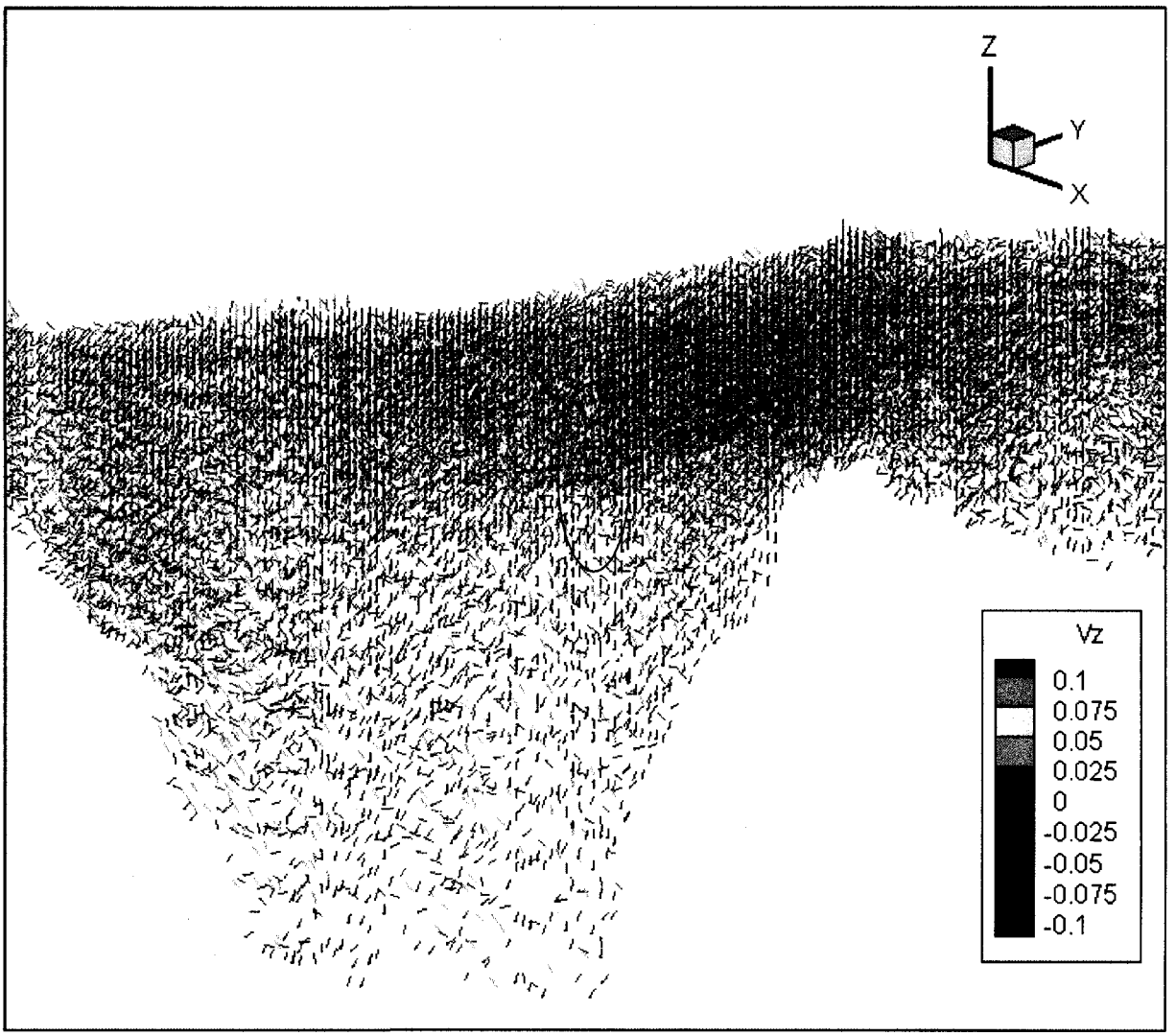

Figure 4-9 Zoomed in 3D color scaled image of interpolated velocity component in z direction (m/s). Flow is from right to left. The vertical scale is 100 times the horizontal scale. The eroding bank is marked with the ellipse.

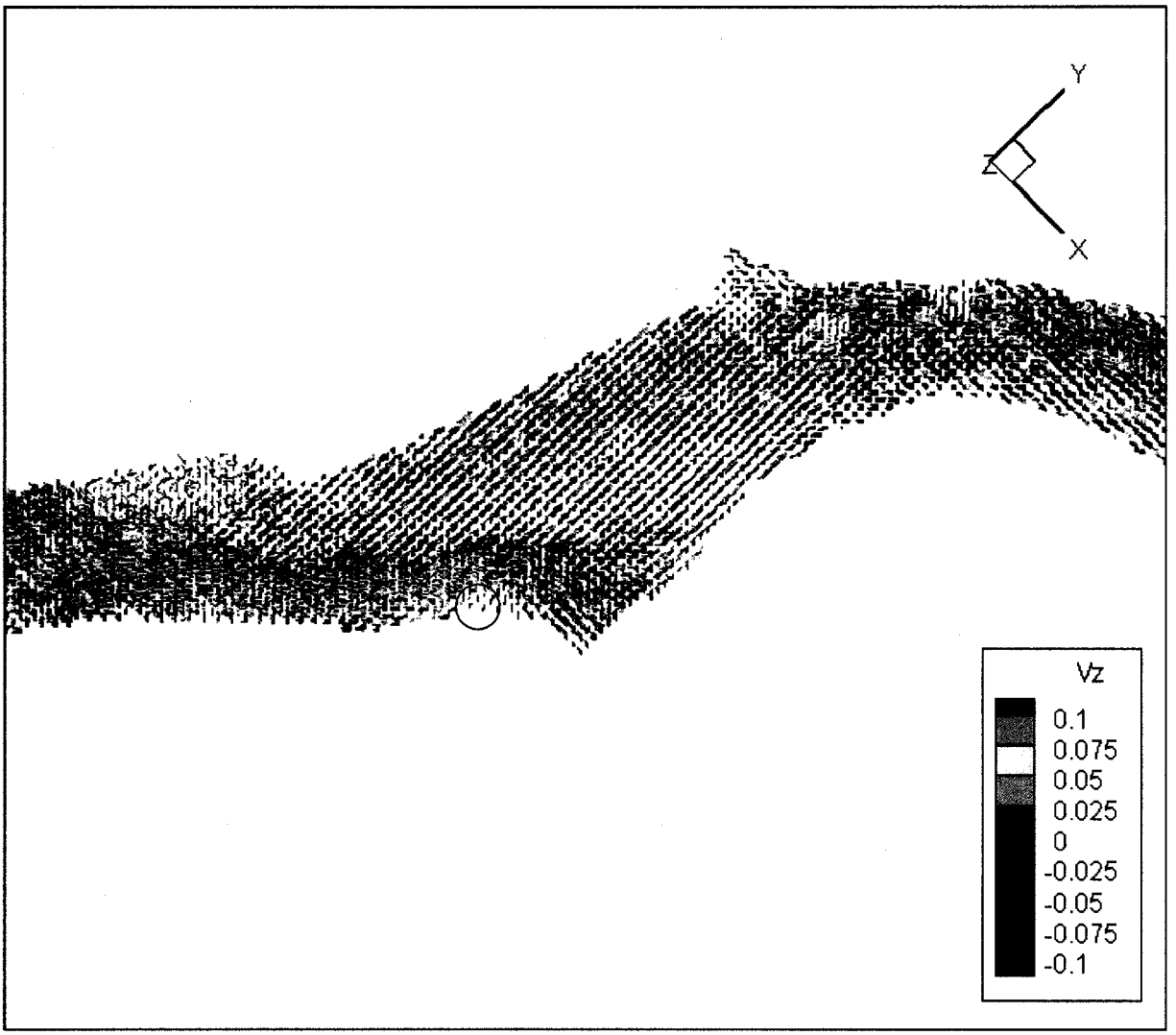

Figure 4-10 Zoomed in color scaled top view of interpolated velocity component in z direction (m/s). Flow is from right to left. The vertical scale is 100 times the horizontal scale. The eroding bank is marked with the circle.

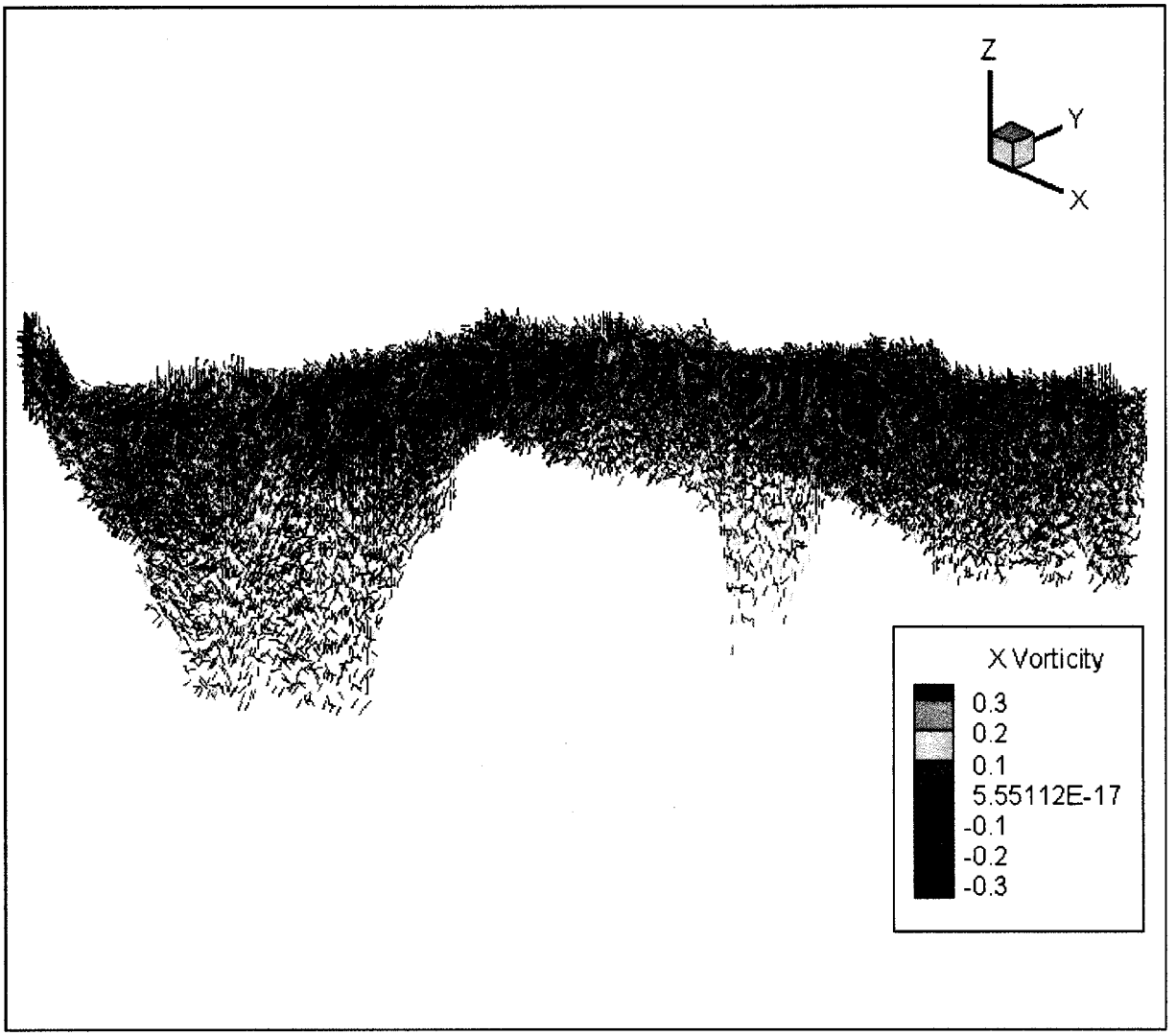

Figure 4-11 3D color scaled image of interpolated vorticity component about x axis (s^{-1}). Flow is from right to left. The vertical scale is 100 times the horizontal scale. The eroding bank is marked with the ellipse.

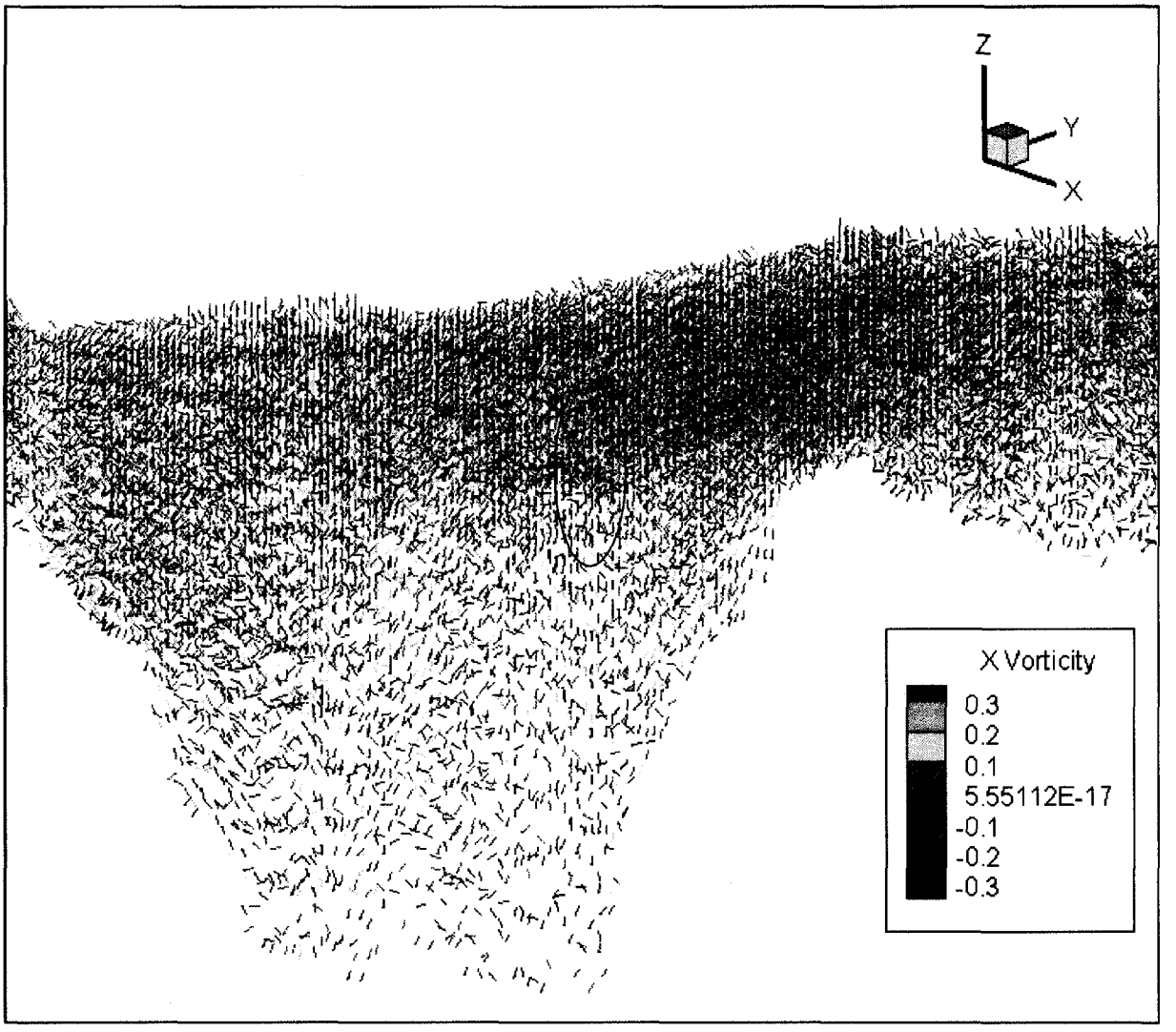

Figure 4-12 Zoomed in 3D color scaled image of interpolated vorticity component about x axis (s^{-1}). Flow is from right to left. The vertical scale is 100 times the horizontal scale. The eroding bank is marked with the ellipse.

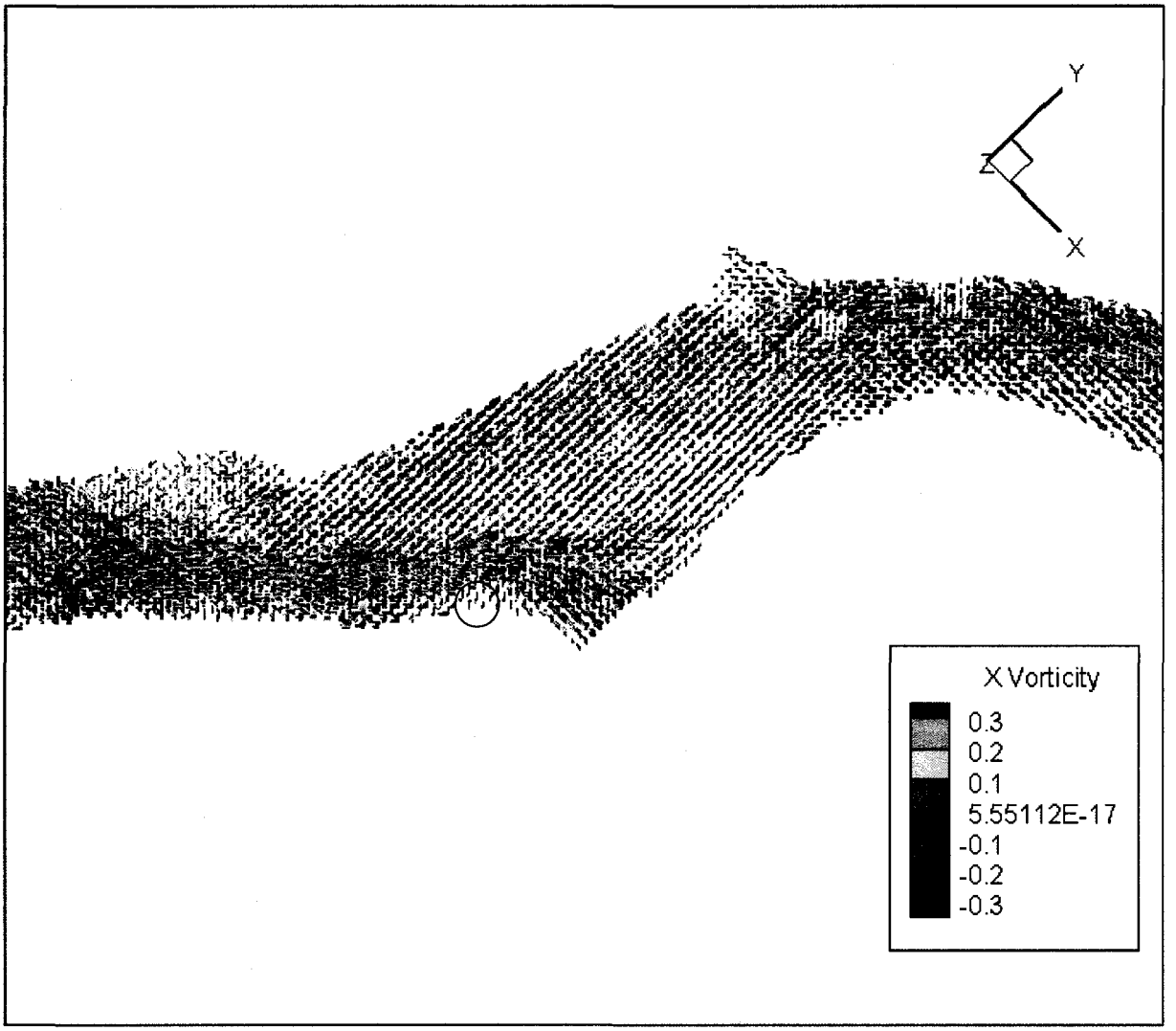

Figure 4-13 Zoomed in color scaled top view of interpolated vorticity component about x axis (s^{-1}). Flow is from right to left. The vertical scale is 100 times the horizontal scale. The eroding bank is marked with the circle.

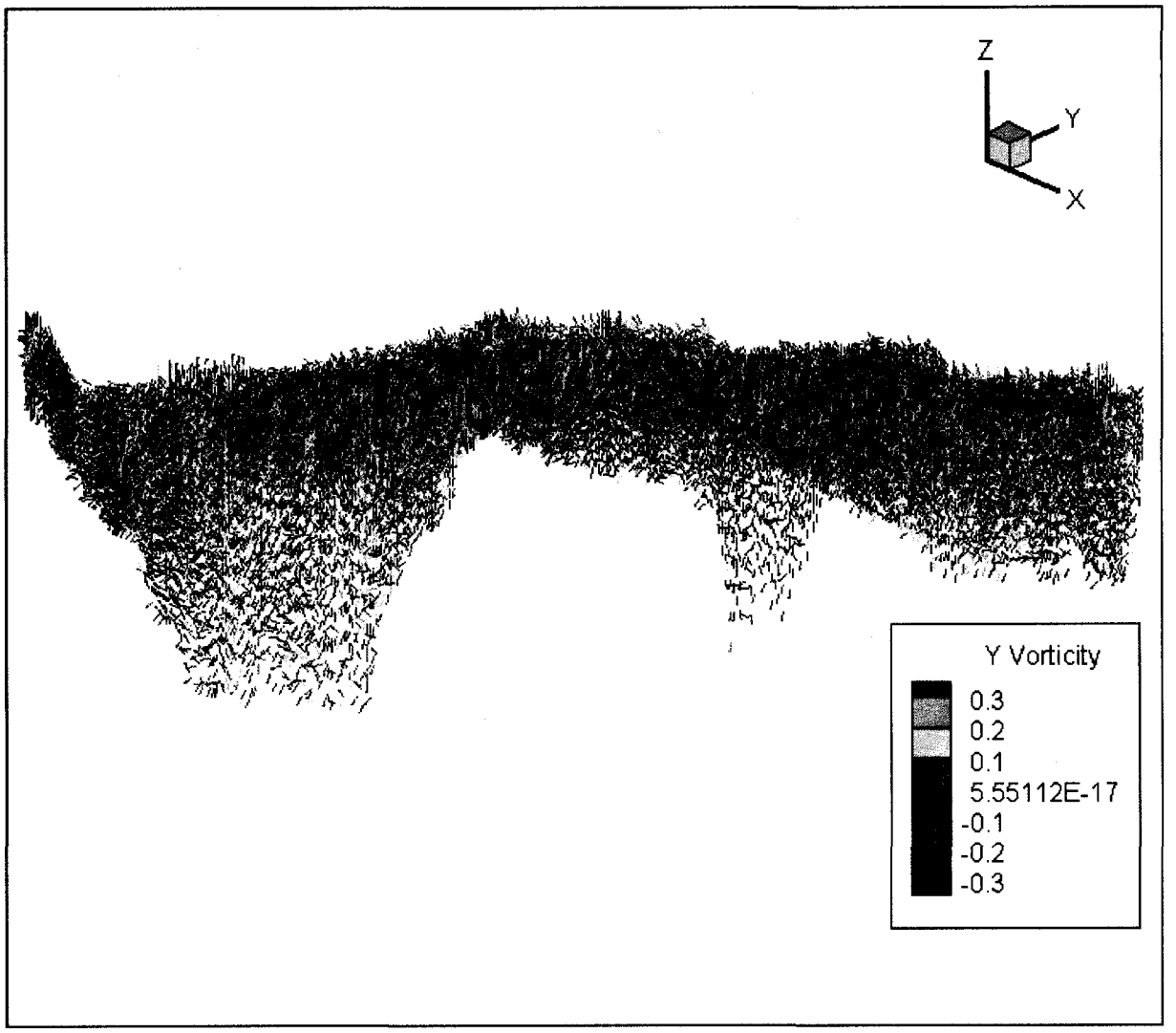

Figure 4-14 3D color scaled image of interpolated vorticity component about y axis (s^{-1}). Flow is from right to left. The vertical scale is 100 times the horizontal scale. The eroding bank is marked with the ellipse.

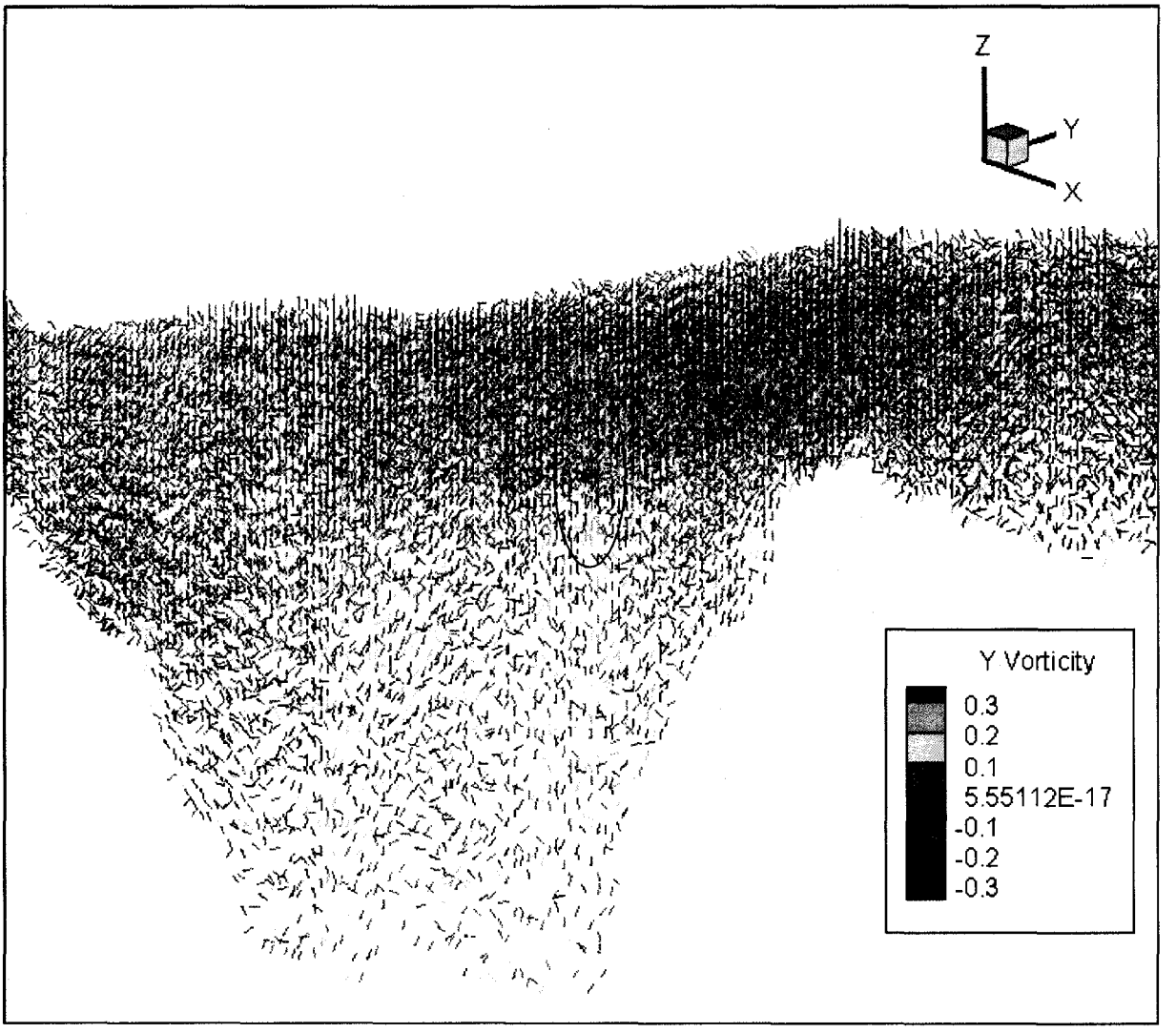

Figure 4-15 Zoomed in 3D color scaled image of interpolated vorticity component about y axis (s^{-1}). Flow is from right to left. The vertical scale is 100 times the horizontal scale. The eroding bank is marked with the ellipse.

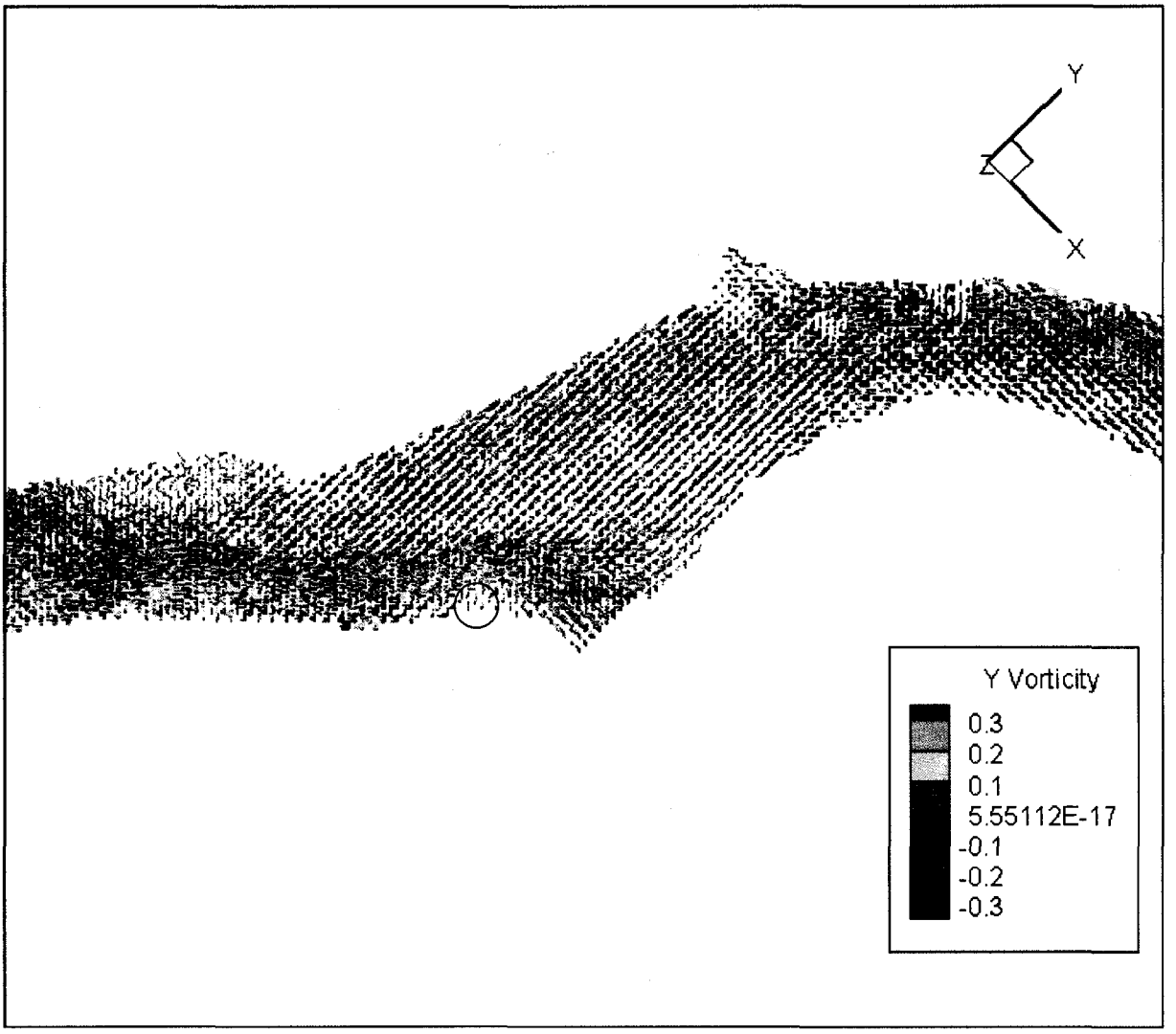

Figure 4-16 Zoomed in color scaled top view of interpolated vorticity component about y axis (s^{-1}). Flow is from right to left. The vertical scale is 100 times the horizontal scale. The eroding bank is marked with the circle.

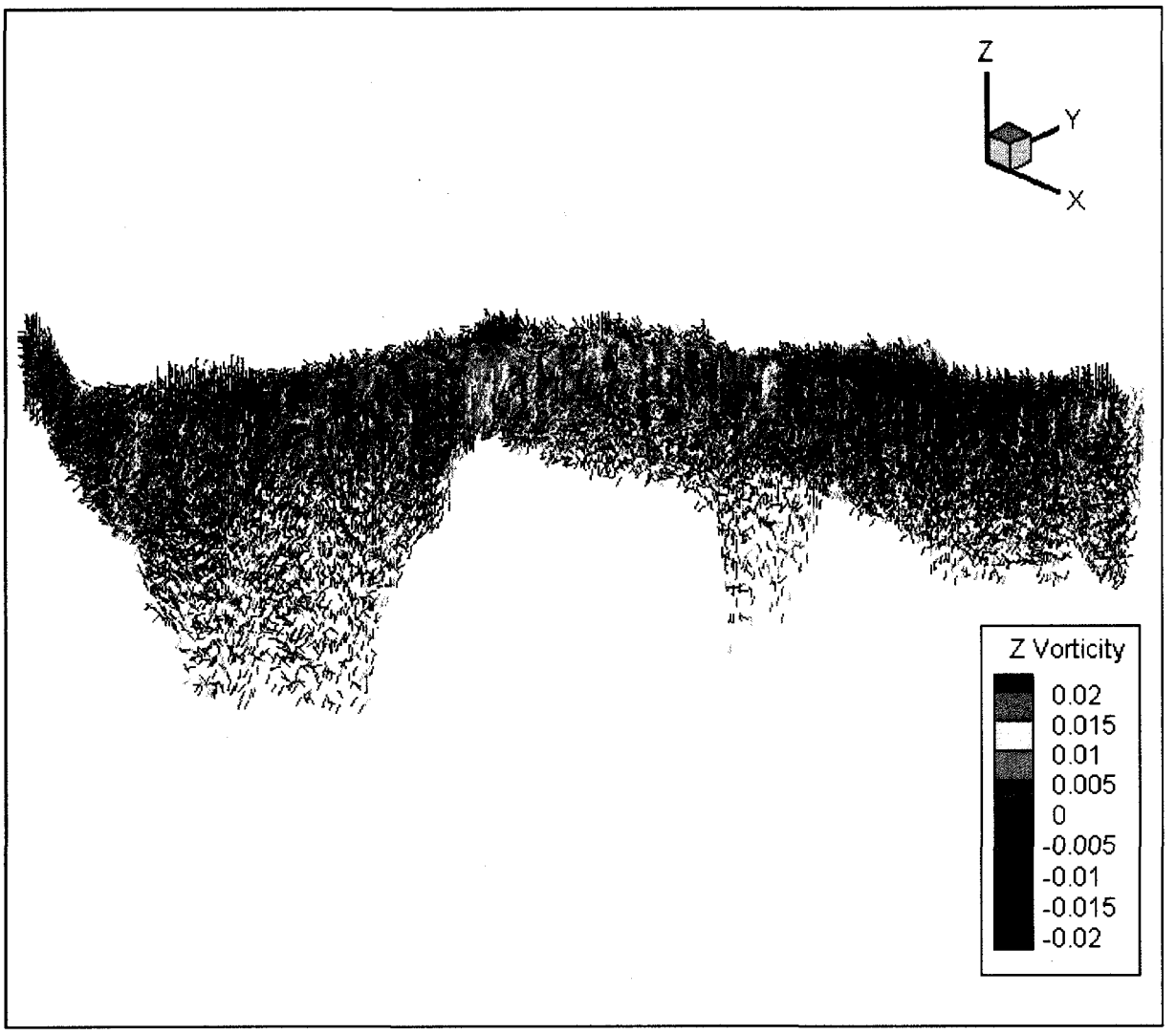

Figure 4-17 3D color scaled image of interpolated vorticity component about z axis (s^{-1}). Flow is from right to left. The vertical scale is 100 times the horizontal scale. The eroding bank is marked with the ellipse.

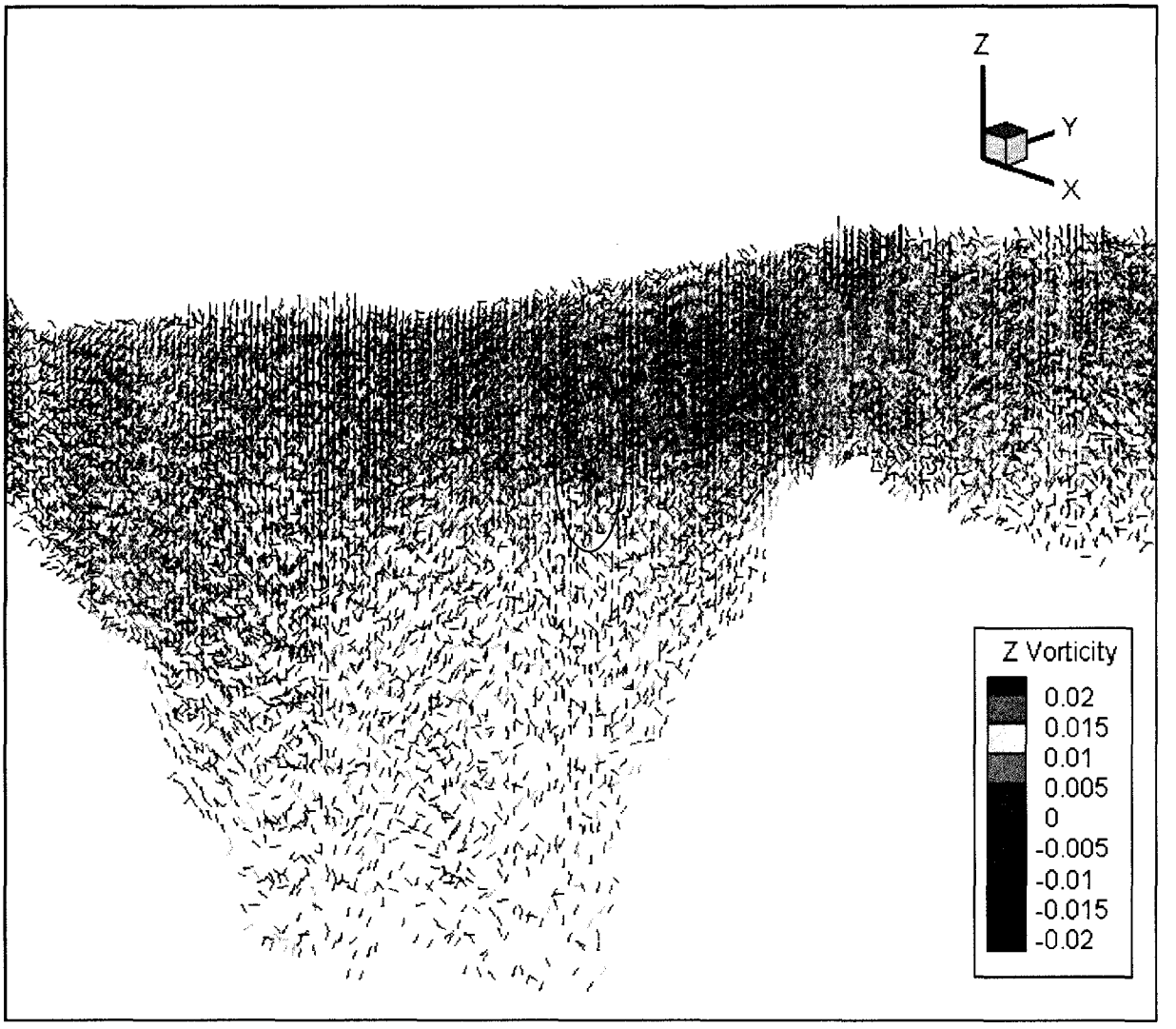

Figure 4-18 Zoomed in 3D color scaled image of interpolated vorticity component about z axis (s^{-1}). Flow is from right to left. The vertical scale is 100 times the horizontal scale. The eroding bank is marked with the ellipse.

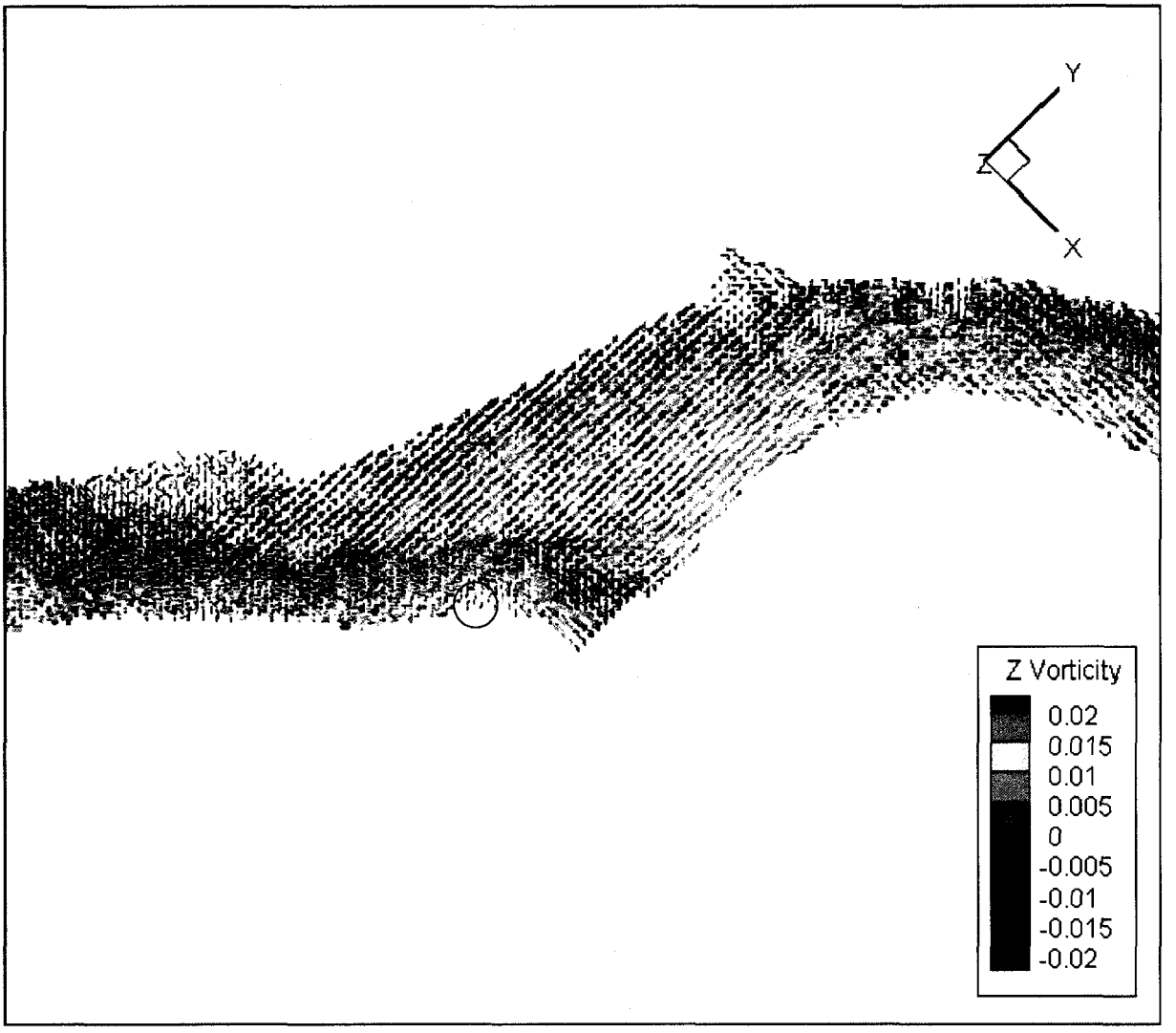

Figure 4-19 Zoomed in color scaled top view of interpolated vorticity component about z axis (s^{-1}). Flow is from right to left. The vertical scale is 100 times the horizontal scale. The eroding bank is marked with the circle.

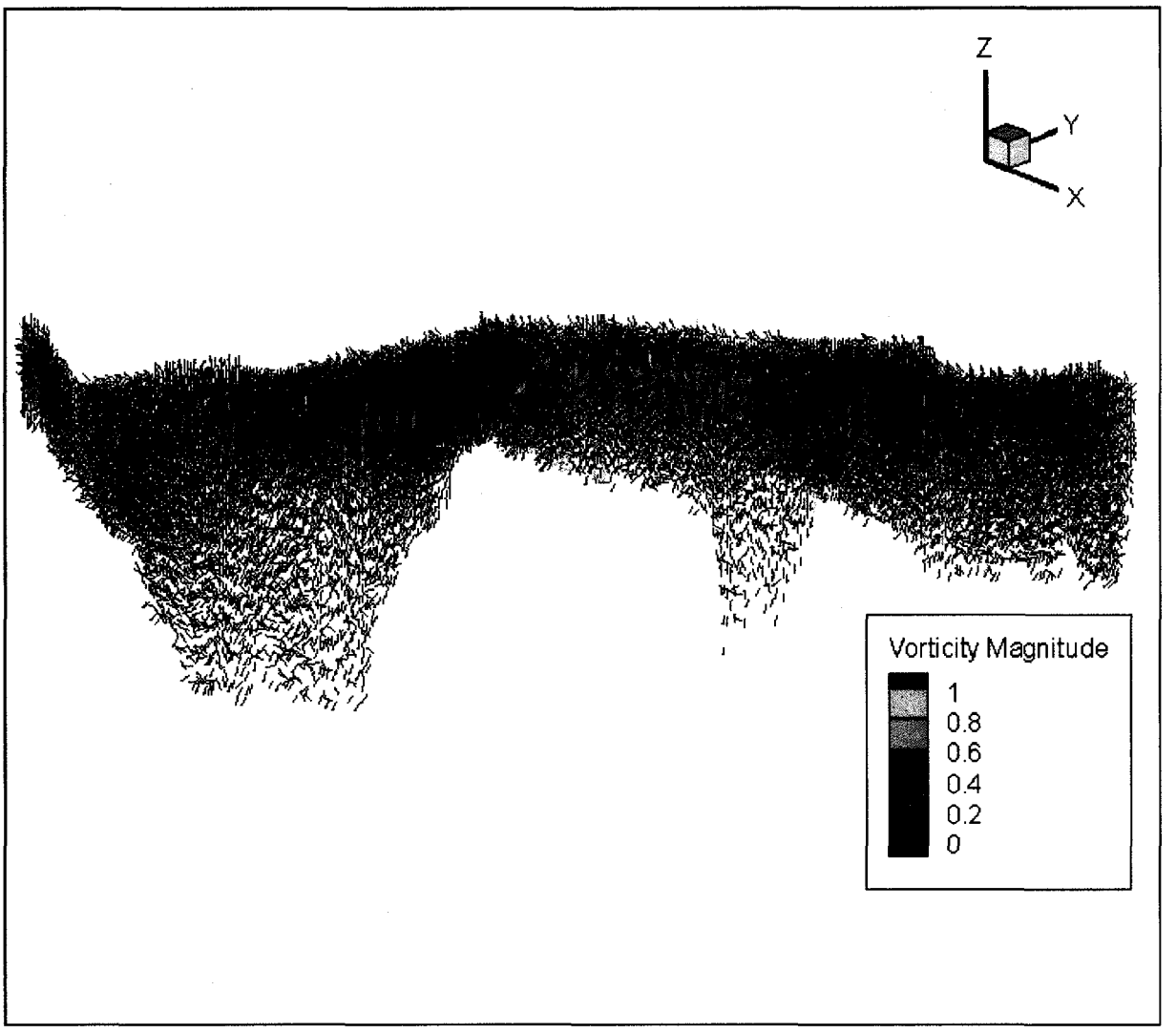

Figure 4-20 3D color scaled image of interpolated vorticity magnitude (s^{-1}). Flow is from right to left. The vertical scale is 100 times the horizontal scale. The eroding bank is marked with the ellipse.

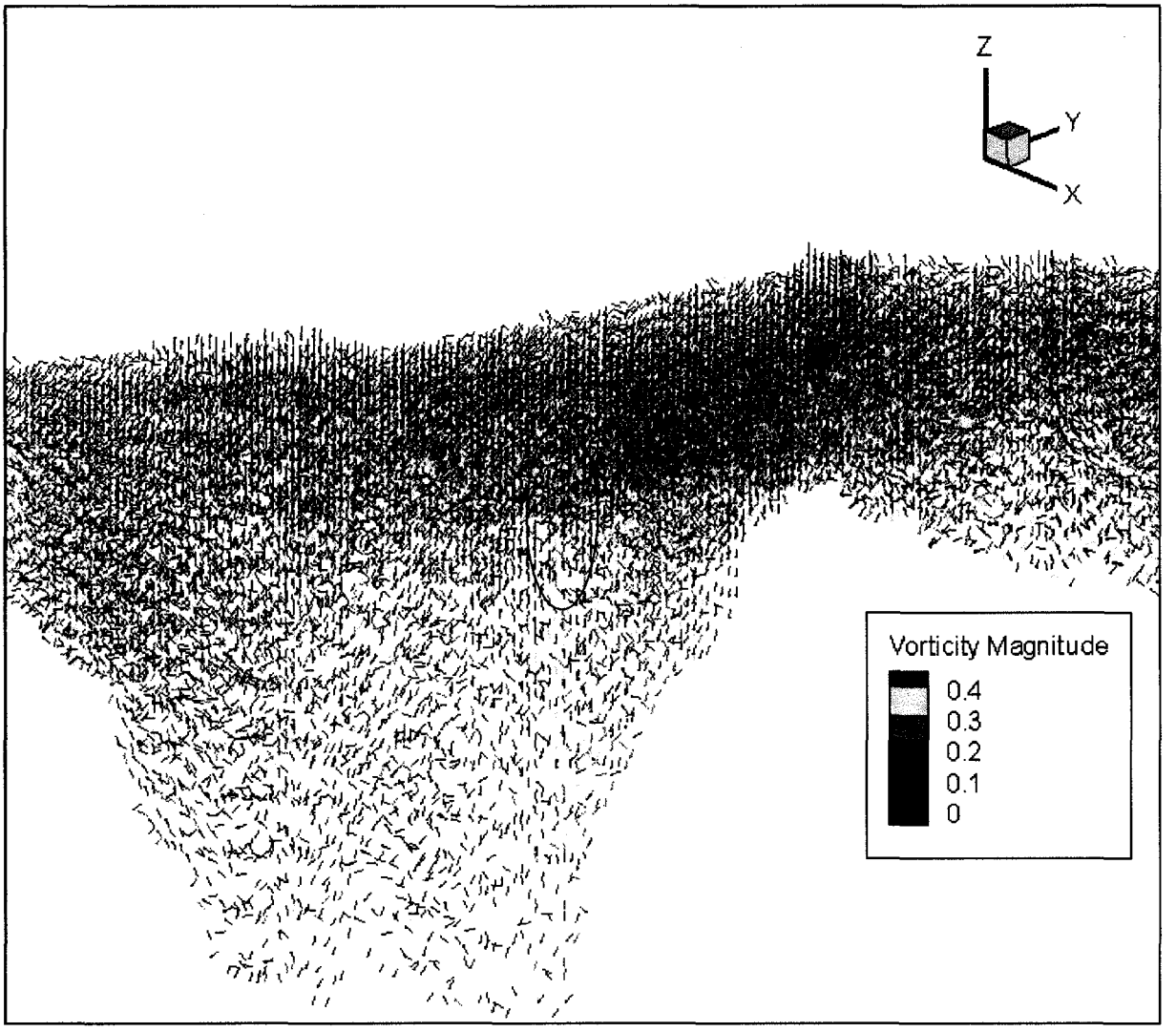

Figure 4-21 Zoomed in 3D color scaled image of interpolated vorticity magnitude (s^{-1}). Flow is from right to left. The vertical scale is 100 times the horizontal scale. The eroding bank is marked with the ellipse.

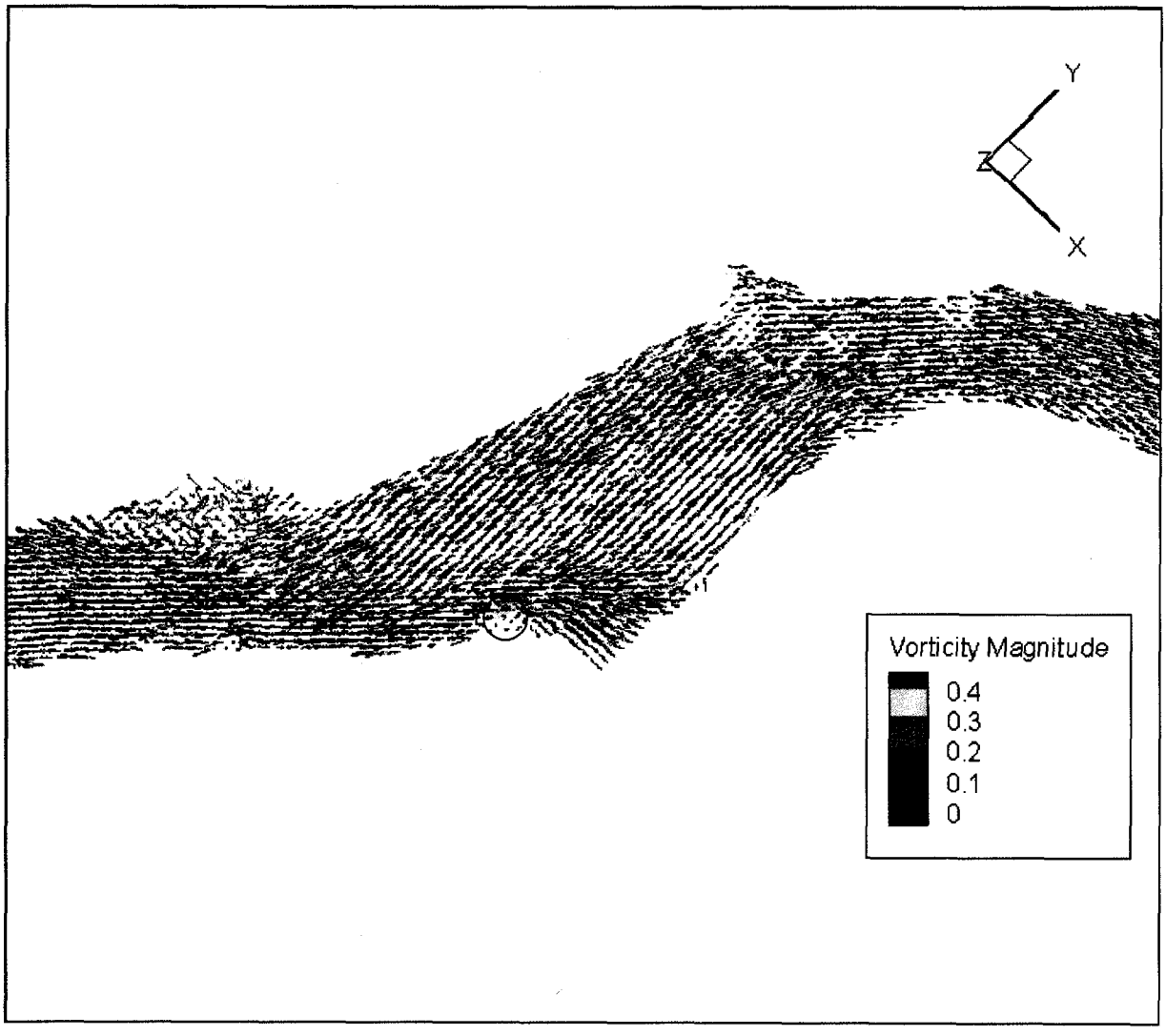

Figure 4-22 Zoomed in color scaled top view of interpolated vorticity magnitude (s^{-1}). Flow is from right to left. The vertical scale is 100 times the horizontal scale. The eroding bank is marked with the circle.

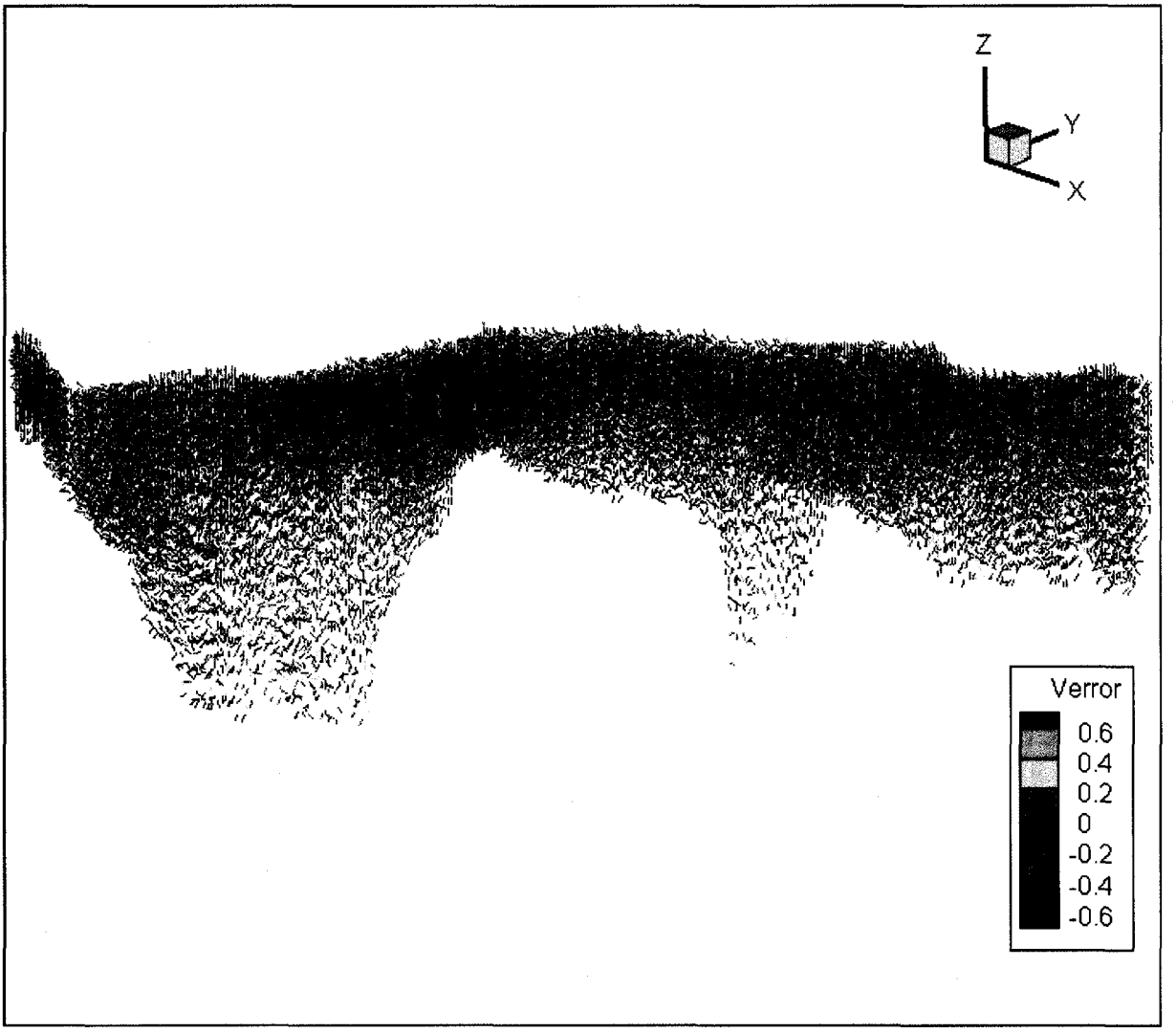

Figure 4-23 3D color scaled image of interpolated error velocity (m/s). Flow is from right to left. The vertical scale is 100 times the horizontal scale.

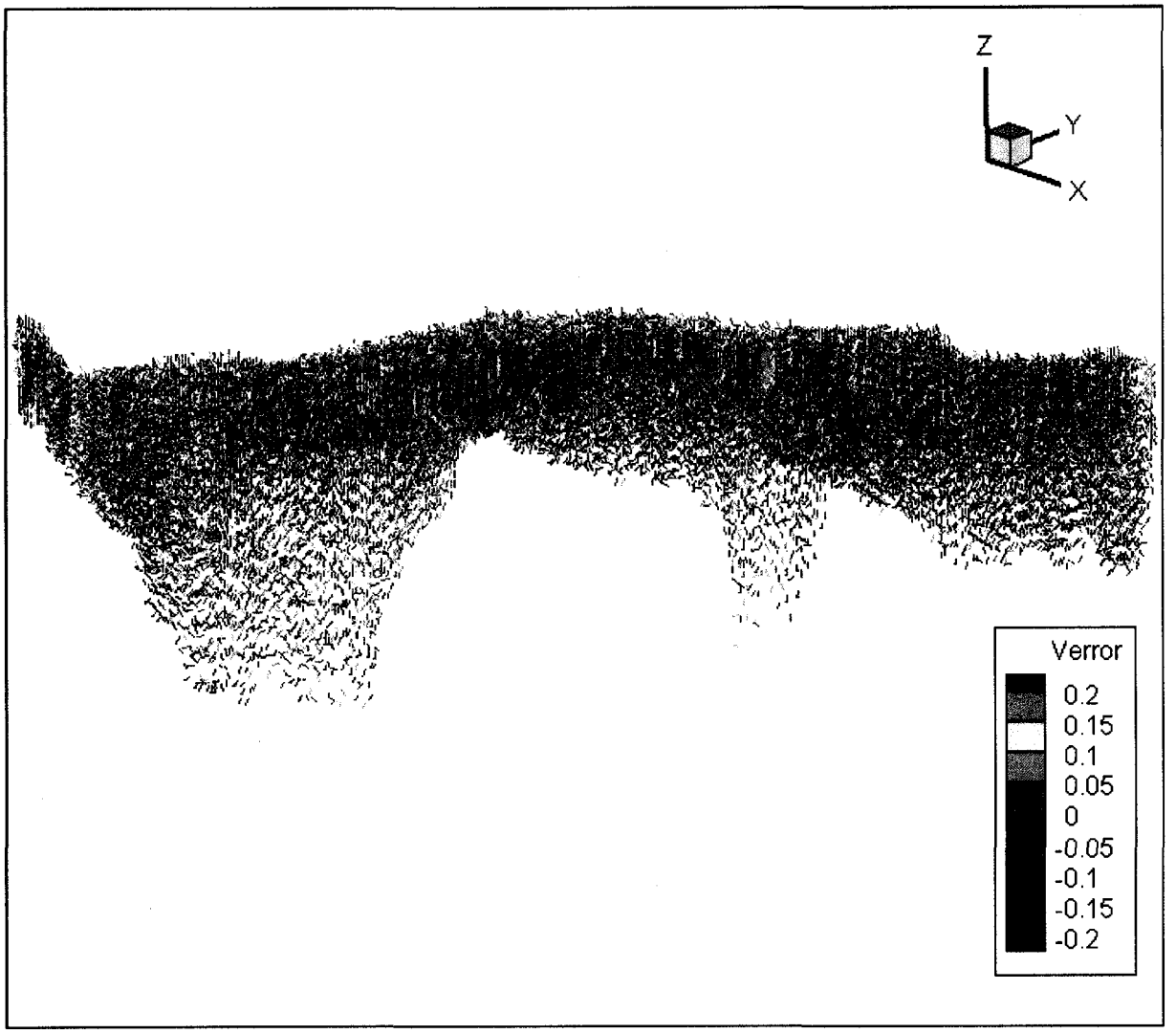

Figure 4-24 3D color scaled image of interpolated error velocity (m/s). Colour scale limited to ± 0.2 m/s. Flow is from right to left. The vertical scale is 100 times the horizontal scale.

5 DISCUSSION

The coherence of the 3D images of river boundary, velocity vector field, error velocity, and vorticity prepared by surveying the reach using a combination of ADCP and GPS on a boat reveals that it is an efficient method for preparing such 3D images in large scales. Interesting flow features revealed that relatively high turbulent vertical velocity, as well as three-dimensional vorticities may be the cause of erosion at the campground.

It is generally believed that high bedload transport occurs where flow velocity and boundary shear stress are high, and so mobile bed numerical models are designed based on this assumption. However, in this reach of Fraser River the highest erosion was observed to occur downstream of the confluence with the Minto Side Channel, where velocity and boundary shear stress calculated by the log-law were relatively low. This was due to the high three-dimensional turbulent flow caused by in inflow of Minto Side Channel to the Fraser River. The fact that three-dimensional flow in complex reaches can govern the sediment transport patterns should be taken into consideration to develop appropriate sediment transport models. The results can also be taken into consideration to deploy methods to divert turbulent flow from the eroding campground bank to slow down or stop its erosion.

It may be interesting to assess whether or not soil properties in the study area had a role in the relatively high erosion rates at the campground. However, natural river banks in the gravel-bed reach of Fraser River are fairly consistently comprised of coarse gravel-cobble material overlain by silt and sand floodplain deposits. Regardless, careful assessment of bank properties in the study reach is warranted.

It is apparent in the vorticity plots (Figures 4-11 to 4-22) that there were locations other than the campground that displayed relatively high vorticity. Many of these locations are at

channel bends where the flow is convectively accelerating around the bend. Vorticity is a measure of the spatial gradient of velocity. For an instantaneous velocity field, vorticity can be generated by turbulent vortices that cause local instantaneous spatial accelerations in the flow. For a mean velocity field, vorticity can be the result of non-uniform but steady convective accelerations (such as secondary currents in bends). Thus, the root cause of calculated vorticity can vary depending on the temporal scale of the flow field.

The interpolated velocity fields generated in this research can be considered a blend of instantaneous and the mean flow field. The data were collected as an asynoptic spatial survey, i.e. irregularly in space over a period of time. It was assumed that the flow was statistically steady during the data collection, based on approximately steady flow measured at the upstream Agassiz gauge. However, turbulent velocity fluctuations associated with macroturbulent eddies that scale with the flow depth would have caused local instantaneous measurements to differ from the mean velocity. Furthermore, single ping ADCP measurements have relatively large errors. Single ping errors can be caused by (Brumley et al. 1991, Simpson 2001):

1) Doppler noise, i.e. imprecision in the estimate of Doppler shift in a range bin. Doppler noise increases with turbulence and shear within the bin, because a non-uniform distribution of scatterer velocities is produced, which reduces the strength of the peak in the Doppler spectrum. For the 1200 kHz Rio Grande ADCP employed in this research, single ping velocity error standard deviation is expected to be 0.14 m/s for conventional Broadband Mode 1 measurements with 25 cm bins (technical specifications provided by RD Instruments), and was calculated to be 0.18 m/s using the RDI software PlanADCP for the ADCP configuration employed in this research.
2) Heterogeneous velocities in a horizontal layer, such that velocities differ between beams. Flow homogeneity is necessary to resolve a three-dimensional Cartesian velocity from multiple beams. The likelihood of flow heterogeneity increases with depth, because beam slant angle causes beams to insonify locations in the flow that are further apart and thus more likely to have different velocities.

3) Employment of an inappropriately small ambiguity (maximum) velocity setting during instrument configuration.
4) Other operator errors, including incorrect compass calibration and erratic or fast boat driving.
5) GPS boat velocity errors, which directly translate to water velocity errors when ADCP velocities are resolved in Earth coordinates. When the radio and satellite communications are made properly and the radios and antennas are not shaded or affected by ferrous material, GPS can provide accurate time and position data. However, natural or man made objects such as mountains, trees, buildings, or bridges can disturb the communications. Therefore, attention should be paid to this issue while analyzing the data. Large GPS errors occurred when GPS communications were shielded near mountains, such as Knob Hill in the middle of the study reach.

The error velocity measured by the ADCP accounts for errors 1, 2, and 3. The fact that error velocities were typically less than ± 0.2 m/s suggests that the measurements met the specifications of the instrument, and horizontal heterogeneity of the velocity field was not a serious problem.

Other researchers have employed multiple passes of individual cross-sections to average out temporal fluctuations associated with macroturbulence and ADCP single ping errors (Dinehart and Bureau 2005). In the present research, greater spatial coverage was achieved using single passes of transects, such that spatial velocity distributions could be mapped. It was assumed that kriging would provide sufficient spatial smoothing to overcome these errors. The spatial distributions were interpolated to 25 m x 25 m grids using kriging. Given that the boat travelled with an average velocity of 1.4 m/s and collected data at approximately 2 Hz, measurements occurred approximately every 0.7 m across a transect, and approximately 36 local values were available within a grid cell if it fell upon a transect. If single ping velocity errors were of the order 0.2 m/s, then velocity error for a 36 ping average should have been $0.2/\sqrt{36} = 0.03$ m/s. Furthermore, flow depth in the reach was about 10 m. If macroturbulent fluctuations had longitudinal length scales equal to about 4 times the

flow depth (Roy et al. 2004), and advected at average flow speeds of 2.5 m/s, then time scales should have been on the order of 16 s, whereas the boat spent approximately 18 s within a 25 m grid cell. Thus, macroturbulent fluctuations may also have been averaged within a grid cell during the interpolation procedure, although it is possible that a single macroturbulent eddy may have dominated the interpolated velocity in a single grid cell. The interpolated velocity field between transect lines is obviously less reliable than in locations with measured data, but should be less subject to macroturbulent velocity fluctuations, because it will be estimated using both adjacent transects. Future research should quantify the contribution of macroturbulent fluctuations and ADCP single ping errors to uncertainty in the interpolated spatial distributions.

Overall, the interpolated velocity field should have represented the mean velocity field, but individual cells may have been dominated by relatively instantaneous measurements of turbulent velocity fluctuations. The vorticity calculated from this velocity field should thus mostly reflect spatial velocity gradients associated with convective acceleration, but may also include relatively instantaneous vorticity associated with turbulent vortices. It is hypothesized that the vorticity calculated in the vicinity of the campground is mostly associated with turbulent vortices. An important distinction of the campground location was it displayed both high three-dimensional vorticity and high oscillating vertical velocity. This lends credence to the suggestion that turbulent vortices were the cause of the high calculated vorticity. Such three-dimensional vortices would cause high near-bank spatial velocity gradients and associated high shear stresses for mobilization of bed and bank sediments. The two-dimensional log-law approach to estimate bed shear stress can not account for these mechanisms, because it depends only the vertical gradient of streamwise velocity, which was relatively low in the vicinity of the campground. Future research should assess the spatial relation between three-dimensional vorticity and vertical velocity throughout the reach. The correlation between these flow variables and observed sediment transport and bank erosion should also be assessed throughout the reach. Such an assessment may further justify the conclusion that highly turbulent three-dimensional vortices caused scour and bank erosion in the campground area.

As discussed above, the data were collected over a period of time and do not exactly represent a snapshot of the flow. This requires the assumption of steady discharge in order for the flow field to maintain statistical stationarity. Reduction of data collection time can increase the accuracy of the results and bring them closer to the assumption of representing a snapshot of the flow. In future research, deployment of several ADCPs in the study area for simultaneous collection of data will reduce the data collection time and bring the results closer to the snapshot assumption. A more intensive survey of the reach is another vision for further research to prepare more accurate images of the boundary, vorticity, and velocity vectors for more detailed analysis of the flow in the reach. Collecting data at the same section in different times and comparing and averaging the results will also provide a better understanding of the flow pattern and its changes over time. Although the flow in the river can be assumed to be statistically steady for a short period of time and a constant discharge, macroturbulences cause the local instantaneous velocities to vary from the local means. Statistical analysis of the flow and study of the role of turbulence will provide better understanding of the flow and its temporal patterns.

6 CONCLUSIONS

ADCP survey of 5.5 km reach of Fraser River can provide coherent three-dimensional images of boundary, flow velocity, error velocity, and vorticity. This is the first time that 3D images of a river boundary, velocity and vorticity are prepared in this scale. This method and such images can be used to better understand the river morphology, the cause of erosion and deposition in rivers, and to improve computer modelling efforts. Data were collected in a relatively short time of two days. The error caused by vertical boat velocity is calculated and removed using RTK-GPS time and location data. The error velocity was reasonably low.

The images reveal that the maximum velocity follows the thalweg. However, the highest sediment transport rate and erosion occurs on the campground eroding bank and not on the thalweg where the shear stress is the maximum. This erosion is hypothesized to be due to relatively high observed vertical velocity as well as the turbulence and three-dimensional vortices caused by the inflow of the Minto Side Channel to the Fraser River and flow separation after the armoured nickpoint.

7 REFRENCES

Brumley, B. H., Cabrera, R. G., Deines, K. L., Terray, E. A. (1991). "Performance of a Broad-Band Acoustic Doppler Current Profiler." Journal of Oceanic Engineering., 16(4).

Bunte, K., Potyondy, J. P., Abt, S. R.; "Development of an Improved Bedload Trap for Sampling Gravel and Cobble Bedload in Coarse Mountain Streams." <http://water.usgs.gov/osw/techniques/sediment/sedsurrogate2003workshop/bunte.pdf> (Oct. 25, 2005)

Dietrich, W. E., Smith, J. D. (1983). "Influence of the Point Bar on Flow Through Curved Channels." America Geography Union, 1983

Dietrich, W. E., Smith, J. D., Dunne, T. (1979). "Flow and Sediment Transport in a Sand Bedded Meander." Journal of Geology, 1979, Vol. 87, p. 305-315.

Dinehart, R.L., Burau, J.R. (2005), Averaged indicators of secondary flow in repeated acoustic Doppler current profiler crossings of bends, Water Resources Research, 41, W09405, doi:10.1029/2005WR04050.

Gaeuman, D., Jacobson, (2006). "Acoustic Bed Velocity and Bedload Dynamics in a Large Sand-Bed River." Journal of Geophysical Research, 111, JF02005, doi:10.1029/2005JF000411, 2006

Helley, E. J., Smith, W. (1971). "Development and Calibration of a Pressure Difference

Bedload Sampler." US Geological Survey, Menlo Park, CA.

Klingeman, P. C., Milhous, R. T. (1971). "Oak Creek Vortex Bedload Sampler." EOS, Transactions, American Geophysical Union, 52(5), 434.

Knighton, D. (1998). "Fluvial forms and Processes." Oxford University Press, Inc., New York

Kostaschuk, R., Best, J., Villard, P., Peakall, J., Franklin M. (2005). "Measuring Flow Velocity and Sediment Transport with an Acoustic Doppler Current Profiler." Geomorphology 68 (2005) 25-37

Minor, B. "Barbs for River Bend Bank Protection: Application of a Three-Dimensional Numerical Model." University of Ottawa, 2006

NovAtel (2005). "OEMO Family Installation and Operation Manual, Rev. 19."

Raudkivi, A. J. (1998). "Loose Boundary Hydraulics." A. A. Balkema, Rotterdam, Netherlands

Rennie, C. D. " Spatial distributions of shear and sediment transport in a large wandering gravel-bed river reach." Presentation to the University of Montreal, October 2006

Rennie, C. D., Church, M. (2007) "ADCP Shear Stress and Bedload Transport in a Large Wandering Gravel-Bed River.", 32nd IAHR Congress, 2007, Venice.

Rennie, C. D., Millar, R. G. (2004). "Measurement of the spatial distribution of fluvial bedload transport velocity in both sand and gravel." Earth Surf. Process. Landforms

29. 1173-1193

Rennie, C. D., Rainville, F. (2006). " Case Study of Precision of GPS Differential Correction Strategies: Influence on aDcp Velocity and Discharge Estimates." Journal of Hydraulic Engineering (ASCE), 132(3):225-234.

Roy, A. G., Buffin-Belanger, T., Lamarre, H., and Kirkbride, A. D. (2004). "Size, shape and dynamics of large-scale turbulent flow structures in a gravel-bed river." Journal of Fluid Mechanics, 500, 1-27.

RD Instruments (1996). "Acoustic Doppler Current Profiler, Principles of Operation, A Practical Primer."

RD Instruments. "ADCP™ products from Teledyne RD Instruments, leading manufacturer of acoustic Doppler current Profilers." <http://www.rdinstruments.com/products.html>(Nov. 30, 2005)

Simpson, M.R. (2001). "Discharge Measurements Using a Broad-Band Acoustic Doppler Current Profiler." United States Geological Survey Open File Report 01-1, Sacremento, CA.

Teledyne Technologies Incorporated (2007). "Teledyne RDI Library & Reference Center Glossary." < www.rdinstruments.com/glossary.html> (Oct. 3, 2007)

USGS (2003). "U.S. Geological Survey Open-File Report 03-001." <http://pubs.usgs.gov/of/2003/of03-001/htmldocs/samplers.htm>(Oct. 7, 2005).

Water Survey of Canada. <http://www.wsc.ec.gc.ca/index_e.cfm?cname=main_e.cfm> (June

2007)

APPENDICES

A COLLECTED DATA

A.1 LIST OF NAVIGATION FILES (N FILES)

Fraser2006_queenscalamity_June24_001n.000

Fraser2006_queenscalamity_June24_002n.000

Fraser2006_queenscalamity_June24_003n.000

Fraser2006_queenscalamity_June24_004n.000

Fraser2006_queenscalamity_June24_005n.000

Fraser2006_queenscalamity_June24_006n.000

A.2 LIST OF RAW DATA FILES (R FILES)

Fraser2006_queenscalamity_June24_001r.000

Fraser2006_queenscalamity_June24_002r.000

Fraser2006_queenscalamity_June24_003r.000

Fraser2006_queenscalamity_June24_004r.000

Fraser2006_queenscalamity_June24_005r.000

Fraser2006_queenscalamity_June24_006r.000

A.3 LIST OF CONFIGURATION FILES (W FILES)

Fraser2006_queenscalamity_June24_000w.000

Fraser2006_queenscalamity_June24_002w.000

Fraser2006_queenscalamity_June24_003w.000

Fraser2006_queenscalamity_June24_004w.000

Fraser2006_queenscalamity_June24_005w.000

Fraser2006_queenscalamity_June24_006w.000

A.4 FORMAT OF NAVIGATION FILES

The following shows the format of the navigation files (n files). The $GPGGA lines are in following format:

log header ($GPGGA),
UTC time (hours/minutes/seconds/decimal seconds),
latitude (DDmm.mm),
latitude direction (N=North, S=South)
longitude(DDDmm.mm),
longitude direction (E=East, W=West)
GPS quality (4: high quality; 5: low quality),
number of satellites in use,
horizontal dilution of precision,
antenna altitude above mean sea level,
unit of antenna altitude (M=meters),
(the rest of fields are not used)

A.5 THE NAVIGATION FILE FRASER2006_QUEENSCALAMITY_JUNE24_001N.000

The following shows the first few pages of the navigation file Fraser2006_queenscalamity_June24_001n.000 (To see the whole file refer to the appendix CD). $RDENS lines show the moments when the data of the previous $GPGGA line are sent to the respective r file.

$GPGGA,181311.40,4912.1443318,N,12158.9709670,W,4,05,1.8,10.166,M,,,01,T001*70
$GPVTG,14.299,T,14.299,M,2.299,N,4.258,K*45
$GPGGA,181311.50,4912.1443921,N,12158.9709416,W,4,05,1.8,10.173,M,,,01,T001*77
$GPVTG,15.444,T,15.444,M,2.252,N,4.172,K*49
$GPGGA,181311.60,4912.1444528,N,12158.9709164,W,4,05,1.8,10.192,M,,,01,T001*79
$GPVTG,15.193,T,15.193,M,2.268,N,4.202,K*44
$GPGGA,181311.70,4912.1445114,N,12158.9708877,W,4,05,1.8,10.196,M,,,01,T001*7C
$GPVTG,17.805,T,17.805,M,2.216,N,4.105,K*49
$RDENS,13,4039456,PC
$GPGGA,181311.80,4912.1445680,N,12158.9708556,W,4,05,1.8,10.197,M,,,00,T001*77
$GPVTG,17.805,T,17.805,M,2.216,N,4.105,K*49
$GPGGA,181311.90,4912.1446258,N,12158.9708201,W,4,05,1.8,10.181,M,,,00,T001*76
$GPVTG,21.918,T,21.918,M,2.247,N,4.161,K*4F
$GPGGA,181312.00,4912.1446835,N,12158.9707845,W,4,05,1.8,10.166,M,,,01,T001*70
$GPVTG,22.027,T,22.027,M,2.242,N,4.152,K*4A
$GPGGA,181312.10,4912.1447366,N,12158.9707486,W,4,05,1.8,10.145,M,,,01,T001*7F
$GPVTG,23.840,T,23.840,M,2.092,N,3.876,K*4D
$GPGGA,181312.20,4912.1447920,N,12158.9707129,W,4,05,1.8,10.125,M,,,01,T001*72
$GPVTG,22.912,T,22.912,M,2.168,N,4.015,K*43
$RDENS,14,4039509,PC

$GPGGA,181312.30,4912.1448457,N,12158.9706813,W,4,05,1.8,10.122,M,,,01,T001*77
$GPVTG,21.084,T,21.084,M,2.075,N,3.843,K*42
$GPGGA,181312.40,4912.1449038,N,12158.9706513,W,4,05,1.8,10.112,M,,,01,T001*72
$GPVTG,18.655,T,18.655,M,2.209,N,4.092,K*48
$GPGGA,181312.50,4912.1449634,N,12158.9706264,W,4,05,1.8,10.110,M,,,00,T001*7D
$GPVTG,18.655,T,18.655,M,2.209,N,4.092,K*48
$GPGGA,181312.60,4912.1450200,N,12158.9706046,W,4,05,1.8,10.115,M,,,00,T001*72
$GPVTG,14.179,T,14.179,M,2.104,N,3.897,K*4C
$GPGGA,181312.70,4912.1450797,N,12158.9705870,W,4,05,1.8,10.136,M,,,00,T001*77
$GPVTG,10.905,T,10.905,M,2.191,N,4.058,K*4C
$RDENS,15,4039563,PC
$GPGGA,181312.80,4912.1451395,N,12158.9705714,W,4,05,1.8,10.145,M,,,00,T001*76
$GPVTG,9.705,T,9.705,M,2.188,N,4.052,K*4E
$GPGGA,181312.90,4912.1451996,N,12158.9705565,W,4,05,1.8,10.165,M,,,00,T001*78
$GPVTG,9.255,T,9.255,M,2.193,N,4.061,K*44
$GPGGA,181313.00,4912.1452569,N,12158.9705403,W,4,05,1.8,10.181,M,,,01,T001*75
$GPVTG,10.499,T,10.499,M,2.100,N,3.889,K*47
$GPGGA,181313.10,4912.1453188,N,12158.9705254,W,4,05,1.8,10.192,M,,,01,T001*78
$GPVTG,8.922,T,8.922,M,2.257,N,4.180,K*41
$GPGGA,181313.20,4912.1453785,N,12158.9705063,W,4,05,1.8,10.192,M,,,01,T001*76
$GPVTG,11.860,T,11.860,M,2.201,N,4.076,K*4A
$GPGGA,181313.30,4912.1454363,N,12158.9704872,W,4,05,1.8,10.193,M,,,01,T001*74
$RDENS,16,4039616,PC
$GPVTG,12.208,T,12.208,M,2.128,N,3.941,K*48
$GPGGA,181313.40,4912.1454968,N,12158.9704620,W,4,05,1.8,10.180,M,,,01,T001*79
$GPVTG,15.282,T,15.282,M,2.262,N,4.190,K*46
$GPGGA,181313.50,4912.1455527,N,12158.9704336,W,4,05,1.8,10.164,M,,,00,T001*77
$GPVTG,15.282,T,15.282,M,2.262,N,4.190,K*46
$GPGGA,181313.60,4912.1456077,N,12158.9704071,W,4,05,1.8,10.141,M,,,00,T001*70
$GPVTG,17.533,T,17.533,M,2.076,N,3.846,K*44

$GPGGA,181313.70,4912.1456655,N,12158.9703806,W,4,05,1.8,10.120,M,,,00,T001*7F
$GPVTG,16.751,T,16.751,M,2.175,N,4.029,K*40
$GPGGA,181313.80,4912.1457212,N,12158.9703538,W,4,05,1.8,10.111,M,,,00,T001*74
$GPVTG,17.441,T,17.441,M,2.106,N,3.901,K*40
$RDENS,17,4039669,PC
$GPGGA,181313.90,4912.1457784,N,12158.9703304,W,4,05,1.8,10.105,M,,,00,T001*73
$GPVTG,15.046,T,15.046,M,2.132,N,3.949,K*4B
$GPGGA,181314.00,4912.1458328,N,12158.9703123,W,4,05,1.8,10.117,M,,,01,T001*75
$GPVTG,12.319,T,12.319,M,2.006,N,3.716,K*49
$GPGSA,M,3,01,22,11,14,19,,,,,,,,3.1,1.8,2.6*3D
$GPGGA,181314.10,4912.1458885,N,12158.9702915,W,4,05,1.8,10.119,M,,,01,T001*7A
$GPVTG,13.755,T,13.755,M,2.065,N,3.825,K*43
$GPGGA,181314.20,4912.1459476,N,12158.9702762,W,4,05,1.8,10.135,M,,,01,T001*78
$GPVTG,9.608,T,9.608,M,2.161,N,4.002,K*4C
$GPGGA,181314.30,4912.1460054,N,12158.9702581,W,4,05,1.8,10.136,M,,,01,T001*7B
$GPVTG,11.565,T,11.565,M,2.127,N,3.941,K*47
$RDENS,18,4039722,PC
$GPGGA,181314.40,4912.1460634,N,12158.9702445,W,4,05,1.8,10.161,M,,,01,T001*77
$GPVTG,8.726,T,8.726,M,2.112,N,3.912,K*47
$GPGGA,181314.50,4912.1461280,N,12158.9702291,W,4,05,1.8,10.167,M,,,00,T001*74
$GPVTG,8.726,T,8.726,M,2.112,N,3.912,K*47
$GPGGA,181314.60,4912.1461894,N,12158.9702106,W,4,05,1.8,10.178,M,,,00,T001*7B
$GPVTG,11.157,T,11.157,M,2.254,N,4.176,K*4B
$GPGGA,181314.70,4912.1462505,N,12158.9701890,W,4,05,1.8,10.181,M,,,00,T001*7F
$GPVTG,13.057,T,13.057,M,2.260,N,4.185,K*40
$GPGGA,181314.80,4912.1463146,N,12158.9701657,W,4,05,1.8,10.182,M,,,00,T001*74
$GPVTG,13.353,T,13.353,M,2.375,N,4.399,K*4A
$RDENS,19,4039775,PC
$GPGGA,181314.90,4912.1463762,N,12158.9701386,W,4,05,1.8,10.187,M,,,00,T001*79
$GPVTG,16.070,T,16.070,M,2.309,N,4.276,K*41

$GPGGA,181315.00,4912.1464397,N,12158.9701126,W,4,05,1.8,10.190,M,,,01,T001*77
$GPVTG,15.052,T,15.052,M,2.369,N,4.387,K*48

B MATLAB CODES

All the codes in the following pages are written by Professor Colin Rennie. However, the code KalmanBinaryReaderSOHEIL2 is slightly modified and adjusted for this project by Shalini Kashyap and Soheil Movafagh. Slight modifications have also been made in the codes Fraser2006_allfles_3 and extractfortecplot_3 by Soheil Movafagh.

B.1 THE MATLAB CODE KALMANBINARYREADERSOHEIL2

```
% TO SORT NAVIGATION DATA VELOCITY REFERENCES AND COMPUTE STATISTICS ON
% PERFORMANCE.
%
% Result of a combination of MatLab scripts from:
%
% David S. Mueller (dmueller@usgs.gov)
% U.S. Geological Survey
% Office of Surface Water
%
% DR. COLIN RENNIE (crennie@genie.uottawa.ca)
% Dept. of Civil Engineering
% University of Ottawa
%
% Francois Rainville (francois.rainville@ec.gc.ca)
% Water Survey of Canada
% Environment Canada
%
%
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% RAW ADCP RDI FILES INGEST SEGMENT:
% The program reads the RDI PD0 format raw data as recorded by WinRiver.
% It accounts for the special locations used by WinRiver to store GPS
data.
% It should read standard PD0 formatted files but was written specifically
% for PD0 files created by WinRiver.
%
% Data are stored in Matlab data structures to provide easy and efficient
% transfer of data between functions. For numeric data the arrays are
% preallocated with "nan" (not a number) so that no value is assumed for
bad
% or missing data. Data coded with -32768 are not stored and default to
"nan".
% No filtering of the data is provided, although some units conversion is
% provided for consistency and convience.
%
% All data structures begin with an % upper case letter.
% All variable names begin with a lower case letter.
% If a specific dimension is associated with a variable its name will
% include an underscore "_" followed by the dimension.
% The dimension abrevations used are as follows:
%   cm      - centimeters
%   deg     - degrees
%   degc    - degrees Celsius
%   dm      - decimeters
%   m       - meters
%   mm      - millimeters
%   mmps    - millimeters per second
%   mps     - meters per second
```

```
%   msc     - minutes seconds hundreths of a second
%   pascal  - pascals
%   ppt     - parts per thousand
%   sec     - seconds
%
% Data that vary with depth are stored so that the bin (depth) varies by
% row and the ensemble varies by column. Most other data are stored so
that
% the ensemble varies by row. Data associated with various beams are
stored
% with a 3rd dimension reflecting the beam number.
%

clear all

fullName = input('File Name?: ', 's');
MagVar = input('Magnetic variation value from WinRiver, in degrees?: ');
FileInfo=dir(fullName);
pathName=fileparts(fullName);
[x,y,z]=fileparts(fullName);
inFileName=strcat(y,z);

% Open File
fidWR=fopen(fullName,'r','l');

%% Read Selected Parameters
% Selected portions of the raw data file are read to determine the number
% of ensembles, the number of bins, and the number of beams. These
% variables are used to preallocate the arrays in the data structures used
% to store the data in Matlab. Although preallocation is not required it
% does improve the performance of Matlab.

% Set the position in the file and read the number of bytes per ensemble.
fseek(fidWR,2,'bof');
bytesPerEns=fread(fidWR,1,'uint16');

% Set position in the file and read the number of data types in the file.
fseek(fidWR,1,'cof');
nTypes=fread(fidWR,1,'uint8');
offset=fread(fidWR,1,'uint16');

% Set position in the file and read the number of beams and number of
bins.
fseek(fidWR,offset+8,'bof');
nBeams=fread(fidWR,1,'uint8');
nBins=fread(fidWR,1,'uint8');

% Compute the number of ensembles, this number will be changed if missing
ensembles found
nEnsembles=ceil(FileInfo.bytes/(bytesPerEns+2));
origNEnsembles=nEnsembles;
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Define Data Storage Structure.
% A data storage structure is used to provide an easy and efficient method
% for passing large amounts of data between functions. The arrays within
% the structure are preallocated for efficiency.
% Clear variables to be used.
clear Hdr Inst Cfg Sensor Gps Wt Bt Nmea;

% Data structure for the Binary Header Data
Hdr=struct( 'bytesPerEns', zeros(1,1),...
            'dataOffsets', zeros(1,nTypes),...
            'nDataTypes', zeros(1,1));

% Data structure for variables related to the instrument
Inst=struct('beamAng', zeros(nEnsembles,1),...
            'beams', zeros(nEnsembles,1),...
            'dataType', repmat(blanks(4),nEnsembles,1),...
            'firmVer', zeros(nEnsembles,1),...
            'freq', zeros(nEnsembles,1),...
            'pat', repmat(blanks(7),nEnsembles,1),...
            'resRDI', zeros(1),...
            'sensorCfg', nan(nEnsembles,1),...
            'xducer', repmat(blanks(12),nEnsembles,1));

% Data structure for bottom track direct commands and other configuration
information
BtCfg=struct('ba', nan(nEnsembles,1),...
            'bc', nan(nEnsembles,1),...
            'be_mmps', nan(nEnsembles,1),...
            'be_mps', nan(nEnsembles,1),...
            'bg', nan(nEnsembles,1),...
            'bm', nan(nEnsembles,1),...
            'bp', nan(nEnsembles,1),...
            'bx_dm', nan(nEnsembles,1));

% Data structure for direct commands and other configuration information
Cfg=struct( 'codeReps', nan(1,1),...
            'coordSys', repmat(blanks(5),1,1),...
            'cpuSerNo', nan(1,8),...
            'cq',  nan(1,1),...
            'cx', nan(1,1),...
            'distBin1_cm', nan(1,1),...
            'distBin1_m', nan(1,1),...
            'ea_deg', nan(1,1),...
            'eb_deg', nan(1,1),...
            'ec', repmat(blanks(8),1,1),...
            'ex', repmat(blanks(8),1,1),...
            'ez', repmat(blanks(8),1,1),...
            'headSrc', repmat(blanks(11),1,1),...
```

```
                'lag_cm', nan(1,1),...
                'mapBins', repmat(blanks(3),1,1),...
                'nBeams', nan(1,1),...
                'pitchSrc', repmat(blanks(11),1,1),...
                'refLayEndCell',  nan(1,1),...
                'refLayStrCell', nan(1,1),...
                'rollSrc', repmat(blanks(11),1,1),...
                'salSrc', repmat(blanks(9),1,1),...
                'wm', nan(1,1),...
                'sosSrc', repmat(blanks(11),1,1),...
                'tempSrc', repmat(blanks(11),1,1),...
                'tp_sec', nan(1,1),...
                'use3beam', repmat(blanks(3),1,1),...
                'usePR', repmat(blanks(3),1,1),...
                'wa', nan(1,1),...
                'wb', nan(1,1),...
                'wc', nan(1,1),...
                'we_mmps', nan(1,1),...
                'wf_cm', nan(1,1),...
                'wg_per', nan(1,1),...
                'wj', nan(1,1),...
                'wn', nan(1,1),...
                'wp', nan(1,1),...
                'ws_cm', nan(1,1),...
                'ws_m', nan(1,1),...
                'xdcrDepSrs', repmat(blanks(9),1,1),...
                'xmitPulse_cm', nan(1,1));

% Data structure for data obtained from the various internal and external
sensors.
Sensor=struct(  'ambientTemp', nan(nEnsembles,1),...
                'attitudeTemp', nan(nEnsembles,1),...
                'attitude', nan(nEnsembles,1),...
                'bitTest', nan(nEnsembles,1),...
                'contamSensor', nan(nEnsembles,1),...
                'date', nan(nEnsembles,4),...
                'dateNotY2k', nan(nEnsembles,3),...
                'dateY2k', nan(nEnsembles,4),...
                'errorStatusWord', repmat(blanks(8),[nEnsembles,1,4]),...
                'headingStdDev', nan(nEnsembles,1),...
                'heading_deg', nan(nEnsembles,1),...
                'mpt_msc', nan(nEnsembles,3),...
                'num', nan(nEnsembles,1),...
                'numFact', nan(nEnsembles,1),...
                'orient', repmat(blanks(4),nEnsembles,1),...
                'pitchStdDev', nan(nEnsembles,1),...
                'pitch_deg', nan(nEnsembles,1),...
                'pressureNeg', nan(nEnsembles,1),...
                'pressurePos', nan(nEnsembles,1),...
                'pressureVar_pascal', nan(nEnsembles,1),...
                'pressure_pascal', nan(nEnsembles,1),...
                'rollStdDev_deg', nan(nEnsembles,1),...
                'roll_deg', nan(nEnsembles,1),...
                'salinity_ppt', nan(nEnsembles,1),...
```

```
                'sos_mps', nan(nEnsembles,1),...
                'temperature_degc', nan(nEnsembles,1),...
                'time', nan(nEnsembles,4),...
                'timeY2k', nan(nEnsembles,4),...
                'xdcrDepth_dm', nan(nEnsembles,1),...
                'xmitCurrent', nan(nEnsembles,1),...
                'xmitVoltage', nan(nEnsembles,1));

% Data structure for the water track data. Data are stored in 3-
dimensional
% arrays with the 1st dimension (rows) being the bin number, the second
% dimension (column) being the ensemble index, and the 3rd dimension being
% the beam number.
Wt=struct(  'corr', nan(nBins,nEnsembles,nBeams),...
            'pergd', nan(nBins,nEnsembles,nBeams),...
            'rssi', nan(nBins,nEnsembles,nBeams),...
            'vel_mps', nan(nBeams,nEnsembles,nBins),...   %this changed to
extract east and north velocity
            'velE_mps', nan(nEnsembles,nBins),...
            'velN_mps', nan(nEnsembles,nBins));
% Data structure for the bottom track data. Data are stored in 2-
dimensional
% arrays with the 1st dimension (rows) being the beam number and the 2nd
% dimension being the ensemble index.
Bt=struct(  'corr', nan(nBeams,nEnsembles),...
            'depth_m', nan(nBeams,nEnsembles),...
            'evalAmp', nan(nBeams,nEnsembles),...
            'extDepth_cm', nan(nEnsembles,1),...
            'pergd', nan(nBeams,nEnsembles),...
            'rssi', nan(nBeams,nEnsembles),...
            'vel_mps', nan(nBeams,nEnsembles),...
            'velE_mps', nan(nEnsembles,1),...
            'velN_mps', nan(nEnsembles,1),...
            'velUp_mps', nan(nEnsembles,1),...
            'velError_mps', nan(nEnsembles,1),...
            'delt', nan(nEnsembles,1));

% Data structure for GPS data: Some variables now defined (eg. vtgTrue).
% Some variables were not initialy pre-defined in DM structure, such as:
% ggaUTC, ggaLat, ggaLon, vtgTrue, vtgMagN and vtgSpd.
Gps =struct( 'alt_m', nan(nEnsembles,1),...
            'ggaDiff', nan(nEnsembles,1),...
            'ggaHdop', nan(nEnsembles,1),...
            'ggaNStats', nan(nEnsembles,1),...
            'ggaVelERead_mps', nan(nEnsembles,1),...        %ggaVel read
from WinRiver binary file, these were calculated by WinRiver and
            'ggaVelNRead_mps', nan(nEnsembles,1),...        %THESE VALUES
SHOULD NOT BE USED IN KALMAN FILTER
            'ggaVelECalc_mps', nan(nEnsembles,1),...        %calculated
gga velocities
            'ggaVelNCalc_mps', nan(nEnsembles,1),...        %THESE ARE THE
GGA VELOCITIES FED INTO KALMAN FILTER
            'gsaPdop', nan(nEnsembles,1),...
            'gsaSat', nan(nEnsembles,6),...
```

```
            'gsaVdop', nan(nEnsembles,1),...
            'lat_deg', nan(nEnsembles,1),...            %Latitude in DD.dd
(degrees, decimal degrees)
            'long_deg', nan(nEnsembles,1),...           %Longitude in
DD.dd (degrees, decimal degrees)
            'ggaLat', nan(nEnsembles,1),...             %Latitude in
DDmm.mm (degrees, minutes, decimal minutes)
            'ggaLon', nan(nEnsembles,1),...             %Longitude in
DDmm.mm (degrees, minutes, decimal minutes)
            'gpsType', nan(nEnsembles,1),...            %Type of Gps
(0=stand alone, 1=fixed, 2=CDGPS, 4=RTK fixed, 5= RTK float, 9=waas)
            'vtgVelE_mps', nan(nEnsembles,1),...        %vtg velocities
stored in *r.* file (these are calculated by WinRiver)
            'vtgVelN_mps', nan(nEnsembles,1),...
            'vtgE_mps', nan(nEnsembles,1),...           %vtg velocities
calculated from vtgSpd and vtgTrue where vtgSpd is taken from VTG Nmea
string
            'vtgN_mps', nan(nEnsembles,1),...           %ONLY vtgE_mps AND
vtgN_mps SHOULD BE USED IN KALMAN FILTER.  vtgVelE_mps and vtgVelN_mps
should not be used
            'posE', nan(nEnsembles,1),...               %gga position east
(m)
            'posN', nan(nEnsembles,1),...               %gga position
north (m)
            'ggaUTC', nan(nEnsembles,1),...             %Gps timestamp
            'vtgTrue', nan(nEnsembles,1),...            %true heading
degrees
            'vtgMagN', nan(nEnsembles,1),...            %magnetic heading
degrees
            'vtgSpd', nan(nEnsembles,1),...             %speed
            'delt', nan(1,nEnsembles),...               %change in time,
normally calculated from GPS ggaUTC timestamp
            'delx', nan(nEnsembles,1),...               %change in gga
position
            'dely', nan(nEnsembles,1),...
            'deltAlt', nan(nEnsembles,1));              %change in antenna
altitude

            % Data structure for Secondary GPS data: This data read
% from ASCII file.
GpsSecondary = struct(  'ggaUTC', nan(nEnsembles,1),...             %GPS
time, units are hhmmss.sss, this is matched to Gps.ggaUTC
                        'lat_deg', nan(nEnsembles,1),...
%Latitude in DD.dd (degrees, decimal degrees)
                        'lat_hem', nan(nEnsembles,1),...
                        'long_deg', nan(nEnsembles,1),...
%Longitude in DD.dd (degrees, decimal degrees)
                        'long_hem', nan(nEnsembles,1),...
                        'ggaLat', nan(nEnsembles,1),...
%Latitude in DDmm.mm (degrees, minutes, decimal minutes)
                        'ggaLon', nan(nEnsembles,1),...
%Longitude in DDmm.mm (degrees, minutes, decimal minutes)
                        'gpsType', nan(nEnsembles,1),...            %Type
of Gps (0=stand alone, 1=fixed, 2=CDGPS, 4=RTK fixed, 5= RTK float,
```

```
9=waas)
                        'ggaNStats', nan(nEnsembles,1),...
%number of satellites
                        'ggaHdop', nan(nEnsembles,1),...
%horizontal dilution
                        'alt_m', nan(nEnsembles,1),...
%antenna alitude above mean sea level
                        'altUnits', nan(nEnsembles,1),...
%antenna altitude units
                        'vtgTrue', nan(nEnsembles,1),...            %true
heading degrees
                        'vtgMagN', nan(nEnsembles,1),...
%magnetic heading degrees
                        'vtgSpd', nan(nEnsembles,1),...             %speed
                        'vtgVelE_mps', nan(nEnsembles,1),...        %vtg
velocities read from WinRiver
                        'vtgVelN_mps', nan(nEnsembles,1),...
                        'ggaVelERead_mps', nan(nEnsembles,1),...
%ggaVel read from WinRiver binary file
                        'ggaVelNRead_mps', nan(nEnsembles,1),...
                        'ggaVelECalc_mps', nan(nEnsembles,1),...
%calculated using delx and dely
                        'ggaVelNCalc_mps', nan(nEnsembles,1),...
                        'posE', nan(nEnsembles,1),...               %gga
position
                        'posN', nan(nEnsembles,1),...
                        'delt', nan(1,nEnsembles),...   %change in time
between ensembles
                        'delx', nan(nEnsembles,1),...   %change in gga
position
                        'dely', nan(nEnsembles,1),...
                        'deltAlt', nan(nEnsembles,1));  %change in antenna
altitude

% Data structure for raw NMEA data strings. The strings are stored
% in a character array with rows being the ensemble index and columns
% the characters in the respective string.
Nmea=struct('gga', repmat(blanks(97),nEnsembles,1),...
            'gsa', repmat(blanks(60),nEnsembles,1),...
            'vtg', repmat(blanks(50),nEnsembles,1));

%Next two structures are only created if GPS is of higher frequency than
bottom tracking

antAltitude=nan(nEnsembles,1);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Read Raw Data
% Data in the PD0 format are organized by leader_id. The leader_id is read
% and the appropriate statements executed to read the data defined by the
% leader_id. All data are read and stored, even data that should not
change
% between ensembles. The data are read until the end of file character is
% encountered.
```

```
%
%Need to use 2D missingEnsem array to know which elements to skip in the
read in
%of data.use missingEnsem array
display('Please wait...');
fseek(fidWR,0,'bof');   %from beginning of file
iEns=0;
missingIndex=1;
while ~feof(fidWR)  %While not end of file

%   Read leader_id.  This is read for each datatype, read for each
ensemble
    leader_id=dec2hex(fread(fidWR,1,'uint16'),4);   %read header data

    %Select appropriate code to read data based on leader_id
    switch leader_id

%%  Read Binary Header Data
        case '7F7F'
            fileloc=ftell(fidWR)-2;
            idataTypes=0;

            iEns=iEns+1;

            Hdr.bytesPerEns(1)=fread(fidWR,1,'uint16');
            fseek(fidWR,1,'cof');
            Hdr.nDataTypes(1)=fread(fidWR,1,'uint8');

Hdr.dataOffsets(1,1:Hdr.nDataTypes(1))=fread(fidWR,Hdr.nDataTypes(1),'uint
16');
             if (idataTypes+1)<=Hdr.nDataTypes(1)

fseek(fidWR,(Hdr.dataOffsets(1,idataTypes+1)+fileloc),'bof');
            else
                fseek(fidWR,fileloc+bytesPerEns-2);
            end

%%      Read Binary Fixed Leader Data
        case '0000'
            idataTypes=idataTypes+1;
            Inst.firmVer(iEns)=fread(fidWR,1,'uint8');

Inst.firmVer(iEns)=Inst.firmVer(iEns)+fread(fidWR,1,'uint8')/100;

            %LSB of system configuration used to determine frequency
            bitls=fread(fidWR,1,'uint8');
            bitls=dec2base(bitls,2,8);

            %Based on value of bitls set frequency
            switch base2dec(bitls(6:8),2)
                case 0;     Inst.freq(iEns)=75;
                case 1;     Inst.freq(iEns)=150;
                case 2;     Inst.freq(iEns)=300;
```

```
    case 3;     Inst.freq(iEns)=600;
    case 4;     Inst.freq(iEns)=1200;
    case 5;     Inst.freq(iEns)=2400;
    otherwise;  Inst.freq(iEns)=NaN;
end;

switch base2dec(bitls(5),2)
    case 0;     Inst.pat(iEns,:)='Concave';
    case 1;     Inst.pat(iEns,:)='Convex ';
    otherwise   Inst.pat(iEns,:)='n/a    ';
end;

Inst.sensorCfg(iEns)=base2dec(bitls(3:4),2)+1;

switch base2dec(bitls(2),2)
    case 0;     Inst.xducer(iEns,:)='Not Attached';
    case 1;     Inst.xducer(iEns,:)='Attached    ';
    otherwise;  Inst.xducer(iEns,:)='n/a         ';
end;

switch base2dec(bitls(1),2)
    case 0;     Sensor.orient(iEns,:)='Down';
    case 1;     Sensor.orient(iEns,:)='Up  ';
    otherwise;  Sensor.orient(iEns,:)='n/a ';
end;

bitms=fread(fidWR,1,'uint8');
bitms=dec2base(bitms,2,8);

switch base2dec(bitms(7:8),2)
    case 0;     Inst.beamAng(iEns)=15;
    case 1;     Inst.beamAng(iEns)=20;
    case 2;     Inst.beamAng(iEns)=30;
    case 3;     Inst.beamAng(iEns)=NaN;
    otherwise;  Inst.beamAng(iEns)=NaN;
end;

switch base2dec(bitms(1:4),2)
    case 4
        Inst.beams(iEns)=4;
    case 5
        Inst.beams(iEns)=5;
        Inst.demod(iEns)=1;
    case 15
        Inst.beams(iEns)=5;
        Inst.demod(iEns)=2;
    otherwise
        Inst.beams(iEns)=NaN;
        Inst.demod(iEns)=NaN;
end;

switch fread(fidWR,1,'uint8')
    case 0;     Inst.dataType(iEns,:)='Real';
```

```
    otherwise;  Inst.dataType(iEns,:)='Simu';
end;

fseek(fidWR,1,'cof');
Cfg.nBeams(1)=fread(fidWR,1,'uint8');
Cfg.wn(1)=fread(fidWR,1,'uint8');
Cfg.wp(1)=fread(fidWR,1,'uint16');
Cfg.ws_cm(1)=fread(fidWR,1,'uint16');
Cfg.ws_m(iEns)=Cfg.ws_cm(1)/100;
Cfg.wf_cm(1)=fread(fidWR,1,'uint16');
Cfg.wm(1)=fread(fidWR,1,'uint8');
Cfg.wc(1)=fread(fidWR,1,'uint8');
Cfg.codeReps(1)=fread(fidWR,1,'uint8');
Cfg.wg_per(1)=fread(fidWR,1,'uint8');
Cfg.we_mmps(1)=fread(fidWR,1,'uint16');
Cfg.tp_sec(1,:)=sum(fread(fidWR,3,'uint8').*[60 1 0.01]');
Cfg.ex(1,:)=dec2base(fread(fidWR,1,'uint8'),2,8);


switch base2dec(Cfg.ex(1,4:5),2)
    case 0;     Cfg.coordSys(1,:)='Beam ';
    case 1;     Cfg.coordSys(1,:)='Inst ';
    case 2;     Cfg.coordSys(1,:)='Ship ';
    case 3;     Cfg.coordSys(1,:)='Earth';
    otherwise;  Cfg.coordSys(1,:)='n/a  ';
end;

switch base2dec(Cfg.ex(1,6),2)
    case 0;     Cfg.usePR(1,:)='No ';
    case 1;     Cfg.usePR(1,:)='Yes';
    otherwise;  Cfg.usePR(1,:)='n/a';
end;

switch base2dec(Cfg.ex(1,7),2)
    case 0;     Cfg.use3beam(1,:)='No ';
    case 1;     Cfg.use3beam(1,:)='Yes';
    otherwise;  Cfg.use3beam(1,:)='n/a';
end;

switch base2dec(Cfg.ex(1,8),2)
    case 0;     Cfg.mapBins(1,:)='No ';
    case 1;     Cfg.mapBins(1,:)='Yes';
    otherwise;  Cfg.mapBins(1,:)='n/a';
end;

Cfg.ea_deg(1)=fread(fidWR,1,'int16')*0.01;
Cfg.eb_deg(1)=fread(fidWR,1,'uint16')*0.01;
Cfg.ez(1,:)=dec2base(fread(fidWR,1,'uint8'),2,8);

switch base2dec(Cfg.ez(1,1:2),2)
    case 0;     Cfg.sosSrc(1,:)='Manual EC  ';
    case 1;     Cfg.sosSrc(1,:)='Calculated ';
    case 3;     Cfg.sosSrc(1,:)='SVSS Sensor';
```

```
    otherwise;  Cfg.sosSrc(1,:)='n/a        ';
end

switch base2dec(Cfg.ez(1,3),2)
    case 0;     Cfg.xdcrDepSrc(1,:)='Manual ED';
    case 1;     Cfg.xdcrDepSrc(1,:)='Sensor   ';
    otherwise;  Cfg.xdcrDepSrc(1,:)='n/a      ';
end

switch base2dec(Cfg.ez(1,4),2)
    case 0;     Cfg.headSrc(1,:)='Manual EH  ';
    case 1;     Cfg.headSrc(1,:)='Int. Sensor';
    otherwise;  Cfg.headSrc(1,:)='n/a        ';
end

switch base2dec(Cfg.ez(1,5),2)
    case 0;     Cfg.pitchSrc(1,:)='Manual EP  ';
    case 1;     Cfg.pitchSrc(1,:)='Int. Sensor';
    otherwise;  Cfg.pitchSrc(1,:)='n/a        ';
end

switch base2dec(Cfg.ez(1,6),2)
    case 0;     Cfg.rollSrc(1,:)='Manual ER  ';
    case 1;     Cfg.rollSrc(1,:)='Int. Sensor';
    otherwise;  Cfg.rollSrc(1,:)='n/a        ';
end

switch base2dec(Cfg.ez(1,7),2)
    case 0;     Cfg.salSrc(1,:)='Manual ES';
    case 1;     Cfg.salSrc(1,:)='n/a      ';
    otherwise;  Cfg.salSrc(1,:)='n/a      ';
end

switch base2dec(Cfg.ez(1,8),2)
    case 0;     Cfg.tempSrc(1,:)='Manual ET  ';
    case 1;     Cfg.tempSrc(1,:)='Int. Sensor';
    otherwise;  Cfg.tempSrc(1,:)='n/a        ';
end

Cfg.ec(1,:)=dec2base(fread(fidWR,1,'uint8'),2,8);
Cfg.distBin1_cm(1)=fread(fidWR,1,'uint16');
Cfg.distBin1_m(1)=Cfg.distBin1_cm(1)/100;
Cfg.xmitPulse_cm(1)=fread(fidWR,1,'uint16');
Cfg.refLayStrCell(1)=fread(fidWR,1,'uint8');
Cfg.refLayEndCell(1)=fread(fidWR,1,'uint8');
Cfg.wa(1)=fread(fidWR,1,'uint8');
Cfg.cx(1)=fread(fidWR,1,'uint8');
Cfg.lag_cm(1)=fread(fidWR,1,'uint16');
Cfg.cpuSerNo(1,:)=fread(fidWR,8,'uint8');
Cfg.wb(1)=fread(fidWR,1,'uint8');
Cfg.cq(1)=fread(fidWR,1,'uint8');
if (idataTypes+1)<=Hdr.nDataTypes(1)
```

```
fseek(fidWR,(Hdr.dataOffsets(1,idataTypes+1)+fileloc),'bof');
            else
                fseek(fidWR,fileloc+bytesPerEns-2,'bof');
            end

%%      Read Variable Leader Data
        case '0080'
            idataTypes=idataTypes+1;
            Sensor.num(iEns)=fread(fidWR,1,'uint16');
            Sensor.dateNotY2k(iEns,:)=fread(fidWR,3,'uint8');
            Sensor.time(iEns,:)=fread(fidWR,4,'uint8');
            Sensor.numFact(iEns)=fread(fidWR,1,'uint8');


Sensor.numTot(iEns)=Sensor.num(iEns)+Sensor.numFact(iEns)*65535;

            Sensor.bitTest(iEns)=fread(fidWR,1,'uint16');
            Sensor.sos_mps(iEns)=fread(fidWR,1,'uint16');
            Sensor.xdcrDepth_dm(iEns)=fread(fidWR,1,'uint16');
            Sensor.heading_deg(iEns)=fread(fidWR,1,'uint16')/100;
            Sensor.pitch_deg(iEns)=fread(fidWR,1,'int16')/100;
            Sensor.roll_deg(iEns)=fread(fidWR,1,'int16')/100;
            Sensor.salinity_ppt(iEns)=fread(fidWR,1,'uint16');
            Sensor.temperature_degc(iEns)=fread(fidWR,1,'uint16');
            Sensor.mpt_msc(iEns,:)=fread(fidWR,3,'uint8');
            Sensor.headingStdDev_deg(iEns)=fread(fidWR,1,'uint8');
            Sensor.pitchStdDev_deg(iEns)=fread(fidWR,1,'uint8')/10;
            Sensor.rollStdDev_deg(iEns)=fread(fidWR,1,'uint8')/10;
            Sensor.xmitCurrent(iEns)=fread(fidWR,1,'uint8');
            Sensor.xmitVoltage(iEns)=fread(fidWR,1,'uint8');
            Sensor.ambientTemp(iEns)=fread(fidWR,1,'uint8');
            Sensor.pressurePos(iEns)=fread(fidWR,1,'uint8');
            Sensor.pressureNeg(iEns)=fread(fidWR,1,'uint8');
            Sensor.attitudeTemp(iEns)=fread(fidWR,1,'uint8');
            Sensor.atttitude(iEns)=fread(fidWR,1,'uint8');
            Sensor.contamSensor(iEns)=fread(fidWR,1,'uint8');

Sensor.errorStatusWord(iEns,:,1)=dec2base(fread(fidWR,1,'uint8'),2,8);

Sensor.errorStatusWord(iEns,:,2)=dec2base(fread(fidWR,1,'uint8'),2,8);

Sensor.errorStatusWord(iEns,:,3)=dec2base(fread(fidWR,1,'uint8'),2,8);

Sensor.errorStatusWord(iEns,:,4)=dec2base(fread(fidWR,1,'uint8'),2,8);
            fseek(fidWR,2,'cof');
            Sensor.pressure_pascal(iEns)=fread(fidWR,1,'uint32');
            Sensor.pressureVar_pascal(iEns)=fread(fidWR,1,'uint32');
            fseek(fidWR,1,'cof');
            Sensor.dateY2k(iEns,:)=fread(fidWR,4,'uint8');
            Sensor.timeY2k(iEns,:)=fread(fidWR,4,'uint8');
            Sensor.date(iEns,:)=Sensor.dateNotY2k(iEns);

Sensor.date(iEns,1)=Sensor.dateY2k(iEns,1)*100+Sensor.dateY2k(iEns,2);
            if (idataTypes+1)<=Hdr.nDataTypes(1)
```

```
fseek(fidWR,(Hdr.dataOffsets(1,idataTypes+1)+fileloc),'bof');
            else
                fseek(fidWR,fileloc+bytesPerEns-2,'bof');
            end

%%      Read Velocity Data
        %Only reads relative water velocity here, velocities are relative
to instrument
        %speed, so may need to subtract speed of instrument.
        case '0100'
            idataTypes=idataTypes+1;
            for iBin=1:Cfg.wn
                for iBeam=1:Inst.beams
                    dummy=fread(fidWR,1,'int16');
                    if dummy~=-32768
                        Wt.vel_mps(iBeam,iEns,iBin)=dummy/1000; %This is
current vel relative to instr.
                    end;
                end;
            end;
            if (idataTypes+1)<=Hdr.nDataTypes(1)

fseek(fidWR,(Hdr.dataOffsets(1,idataTypes+1)+fileloc),'bof');
            else
                fseek(fidWR,fileloc+bytesPerEns-2,'bof');
            end
%%      Read Correlation Magnitude
        %This is iBeam correlation for 1st bin of each iBeam
        case '0200'
            idataTypes=idataTypes+1;
            for iBin=1:Cfg.wn
                for iBeam=1:Inst.beams
                    dummy=fread(fidWR,1,'uint8');
                    if dummy~=-32768
                        Wt.corr(iBin,iEns,iBeam)=dummy; %Stores
correlation for 1st bin for each iBeam
                    end;
                end;
            end;
            if (idataTypes+1)<=Hdr.nDataTypes(1)

fseek(fidWR,(Hdr.dataOffsets(1,idataTypes+1)+fileloc),'bof');
            else
                fseek(fidWR,fileloc+bytesPerEns-2,'bof');
            end

%%      Read Echo Intensity
        case '0300'
            idataTypes=idataTypes+1;
            for iBin=1:Cfg.wn
                for iBeam=1:Inst.beams
                    dummy=fread(fidWR,1,'uint8');
                    if dummy~=-32768
```

```
                    Wt.rssi(iBin,iEns,iBeam)=dummy;
                end;
            end;
        end;
        if (idataTypes+1)<=Hdr.nDataTypes(1)

fseek(fidWR,(Hdr.dataOffsets(1,idataTypes+1)+fileloc),'bof');
        else
            fseek(fidWR,fileloc+bytesPerEns-2,'bof');
        end

%%      Read Percent-Good Data
        case '0400'
            idataTypes=idataTypes+1;
            for iBin=1:Cfg.wn
                for iBeam=1:Inst.beams
                    dummy=fread(fidWR,1,'uint8');
                    if dummy~=-32768
                        Wt.pergd(iBin,iEns,iBeam)=dummy;
                    end;
                end;
            end;
            if (idataTypes+1)<=Hdr.nDataTypes(1)

fseek(fidWR,(Hdr.dataOffsets(1,idataTypes+1)+fileloc),'bof');
        else
            fseek(fidWR,fileloc+bytesPerEns-2,'bof');
        end

%%      Read Bottom Track Data
        case '0600'
        idataTypes=idataTypes+1;
            BtCfg.bp(iEns)=fread(fidWR,1,'uint16');          %number of bt
pings to average for each ensemble
            long1=fread(fidWR,1,'uint16');                   %longitude in
binary angular measurement, this is for gga but

            BtCfg.bc(iEns)=fread(fidWR,1,'uint8');           %Minimum
correlation magnitude
            BtCfg.ba(iEns)=fread(fidWR,1,'uint8');           %Minimum
elevation amplitude
            BtCfg.bg(iEns)=fread(fidWR,1,'uint8');           %Minimum bt
pings that must be good to output velocity
            BtCfg.bm(iEns)=fread(fidWR,1,'uint8');           %Bottom
tracking mode
            BtCfg.be_mmps(iEns)=fread(fidWR,1,'uint16');     %Maximum Error
velocity
            BtCfg.be_mps(iEns)=BtCfg.be_mmps(iEns)/1000;
            Gps.lat_deg(iEns,1)=(fread(fidWR,1,'int32')/2^31)*180;
            for iBeam=1:Inst.beams
                    dummy=fread(fidWR,1,'uint16');
                    if dummy~=-32768
                        Bt.depth_m(iBeam,iEns)=dummy/100;
                    end;
```

```
            end;
            for iBeam=1:Inst.beams
                dummy=fread(fidWR,1,'int16');
                if dummy~=-32768
                    Bt.vel_mps(iBeam,iEns)=dummy/1000;      %Value and
direction of velocity depends on co-ordinate system
                end;                                        %See WorkHorse
Man. page 133 bytes 25-32
            end;
            for iBeam=1:Inst.beams
                dummy=fread(fidWR,1,'uint8');
                if dummy~=-32768
                    Bt.corr(iBeam,iEns)=dummy;
                end;
            end;
            for iBeam=1:Inst.beams
                dummy=fread(fidWR,1,'uint8');
                if dummy~=-32768
                    Bt.evalAmp(iBeam,iEns)=dummy;
                end;
            end;
            for iBeam=1:Inst.beams
                dummy=fread(fidWR,1,'uint8');
                if dummy~=-32768
                    Bt.pergd(iBeam,iEns)=dummy;     %Bottom track percent
good data for each beam
                end
            end;
            dummy=fread(fidWR,1,'uint16');
            if dummy~=-32768
                Gps.alt_m(iEns)=(dummy-32768)/10;   %GGA altitude
(m))x10+32768 has been converted back
            end
            long2=fread(fidWR,1,'uint16');

            Gps.long_deg(iEns,1)=((long1+long2*2^16)/2^31)*180;
            if Gps.long_deg(iEns,1) > 180
                Gps.long_deg(iEns,1)=Gps.long_deg(iEns,1)-360;
            end;
            Bt.extDepth_cm(iEns)=fread(fidWR,1,'int16');        %if Depth
sounder enabled, DBT or DBS depth in cm

            %Reading of Velocities
            dummy=fread(fidWR,1,'int16');
            if dummy~=-32768
                    Gps.ggaVelERead_mps(iEns)=dummy*-1/1000;
%As WinRiver reverses direction, put in negative
            end
%This is read value, calculated value takes priority
            dummy=fread(fidWR,1,'int16');
            if dummy~=-32768
                    Gps.ggaVelNRead_mps(iEns)=dummy*-1/1000;
            end
            dummy=fread(fidWR,1,'int16');
```

```
            if dummy~=-32768
                    Gps.vtgVelE_mps(iEns)=dummy*-1/1000;
            end
            dummy=fread(fidWR,1,'int16');
            if dummy~=-32768
                    Gps.vtgVelN_mps(iEns)=dummy*-1/1000;
            end
            dummy=fread(fidWR,1,'uint8');
            if dummy~=0
                Gps.gsaVdop(iEns)=dummy;
            end
             dummy=fread(fidWR,1,'uint8');
            if dummy~=0
                Gps.gsaPdop(iEns)=dummy;
            end
             dummy=fread(fidWR,1,'uint8');
            if dummy~=0
                Gps.ggaNStats(iEns)=dummy;      %gga number of satellites
            end
            fseek(fidWR,1,'cof');
            Gps.gsaSat(iEns,5)=fread(fidWR,1,'uint8');      %GSA satellite
5
            Gps.gsaSat(iEns,6)=fread(fidWR,1,'uint8');      %GSA satellite
6
            Gps.ggaDiff(iEns)=fread(fidWR,1,'uint8');       %GGA quality
             dummy=fread(fidWR,1,'uint8');
            if dummy~=0
                Gps.ggaHdop(iEns)=dummy/10;
            end
            Gps.gsaSat(iEns,1)=fread(fidWR,1,'uint8');      %GSA satellite
1
            Gps.gsaSat(iEns,2)=fread(fidWR,1,'uint8');      %GSA satellite
2
            Gps.gsaSat(iEns,3)=fread(fidWR,1,'uint8');      %GSA satellite
3
            Gps.gsaSat(iEns,4)=fread(fidWR,1,'uint8');      %GSA satellite
4
            BtCfg.bx_dm(iEns)=fread(fidWR,1,'uint16');      %Maximum
tracking depth
            Bt.rssi(1,iEns)=fread(fidWR,1,'uint8');         %receiver
signal strength indicator, beam 1
            Bt.rssi(2,iEns)=fread(fidWR,1,'uint8');         %receiver
signal strength indicator, beam 2
            Bt.rssi(3,iEns)=fread(fidWR,1,'uint8');         %receiver
signal strength indicator, beam 3
            Bt.rssi(4,iEns)=fread(fidWR,1,'uint8');         %receiver
signal strength indicator, beam 4
            Cfg.wj(iEns)=fread(fidWR,1,'uint8');            %gain level
for shallow water

            for iBeam=1:Inst.beams
                dummy=fread(fidWR,1,'uint8');
                %Depth in m for each beam from sea bottom
                %These are MSB, added to lower or LSB by formula
```

```
                if dummy~=-32768
                    Bt.depth_m(iBeam,iEns)=Bt.depth_m(iBeam,iEns)+...
                    (dummy*2^16)/100;
                end

            end;


            if (idataTypes+1)<=Hdr.nDataTypes(1)

fseek(fidWR,(Hdr.dataOffsets(1,idataTypes+1)+fileloc),'bof');
            else
                fseek(fidWR,fileloc+bytesPerEns-2,'bof');
            end

%%      Read DBT NMEA String
%       Seems to read in entire string
        case '2100'
            idataTypes=idataTypes+1;
            marker=ftell(fidWR);
            fseek(fidWR,(Hdr.dataOffsets(1,idataTypes)+fileloc+2),'bof');
            Nmea.dbt(iEns,:)=char(fread(fidWR,38,'char'));
            dummy=zeros(1,38);
            endstr=find(Nmea.dbt(iEns,:)==char(13));
            dummy(1,1:endstr)=Nmea.dbt(iEns,1:endstr);
            Nmea.dbt(iEns,:)=dummy(1,:);
            if (idataTypes+1)<=Hdr.nDataTypes(1)

fseek(fidWR,(Hdr.dataOffsets(1,idataTypes+1)+fileloc),'bof');
            else
                fseek(fidWR,fileloc+bytesPerEns-2,'bof');
            end

%%      Read GGA NMEA String
        case '2101'
            idataTypes=idataTypes+1;
            marker=ftell(fidWR);
            fseek(fidWR,(Hdr.dataOffsets(1,idataTypes)+fileloc+2),'bof');
            dummy1=char(fread(fidWR,97,'char'));
            dummy=zeros(1,97);

            endstr=find(dummy1==char(42))+2;
            dummy(1,1:endstr)=dummy1(1:endstr)';
            Nmea.gga(iEns,:)=dummy(1,:);

            idx=find(Nmea.gga(iEns,:)==',');
            if numel(idx)~=0
                if idx(1)+1<=idx(2)-1

Gps.ggaUTC(iEns,1)=str2num(Nmea.gga(iEns,idx(1)+1:idx(2)-1));
                else
                    Gps.ggaUTC(iEns,1)=NaN;
                end
```

```
                if idx(2)+1<=idx(3)-1

Gps.ggaLat(iEns,1)=str2num(Nmea.gga(iEns,idx(2)+1:idx(3)-1));

                else
                    Gps.ggaLat(iEns,1)=NaN;
                end
                if idx(4)+1<=idx(5)-1

Gps.ggaLon(iEns,1)=str2num(Nmea.gga(iEns,idx(4)+1:idx(5)-1));

                else
                    Gps.ggaLon(iEns,1)=NaN;
                end
                if idx(6)+1<=idx(7)-1

Gps.gpsType(iEns,1)=str2num(Nmea.gga(iEns,idx(6)+1:idx(7)-1));

                else
                    Gps.gpsType(iEns,1)=NaN;
                end
                if idx(9)+1<=idx(10)-1

antAltitude(iEns,1)=str2num(Nmea.gga(iEns,idx(9)+1:idx(10)-1));
                else
                   antAltitude(iEns,1)=NaN;
                end

            else
                    Gps.ggaUTC(iEns,1)=NaN;
                    Gps.ggaLat(iEns,1)=NaN;
                    Gps.ggaLon(iEns,1)=NaN;
                    Gps.gpsType(iEns,1)=NaN;
            end
            if (idataTypes+1)<=Hdr.nDataTypes(1)

fseek(fidWR,(Hdr.dataOffsets(1,idataTypes+1)+fileloc),'bof');
            else
                fseek(fidWR,fileloc+bytesPerEns-2,'bof');
            end

%%      Read VTG NMEA String
        case '2102'
            idataTypes=idataTypes+1;
            marker=ftell(fidWR);
            fseek(fidWR,(Hdr.dataOffsets(1,idataTypes)+fileloc+2),'bof');
            dummy2=char(fread(fidWR,50,'char'))';
            dummy=zeros(1,50);
            endstr=find(dummy2==char(42))+2;
            dummy(1,1:endstr)=dummy2(1:endstr);
            Nmea.vtg(iEns,:)=dummy(1,:);
            idx=find(Nmea.vtg(iEns,:)==',');
            if numel(idx)~=0
```

```
                if idx(1)+1<=idx(2)-1

Gps.vtgTrue(iEns,1)=str2num(Nmea.vtg(iEns,idx(1)+1:idx(2)-1));
                else
                   Gps.vtgTrue(iEns,1)=NaN;
                end
                if idx(3)+1<=idx(4)-1

Gps.vtgMagN(iEns,1)=str2num(Nmea.vtg(iEns,idx(3)+1:idx(4)-1));
                else
                    Gps.vtgMagN(iEns,1)=NaN;
                end
                if idx(7)+1<=idx(8)-1

Gps.vtgSpd(iEns,1)=str2num(Nmea.vtg(iEns,idx(7)+1:idx(8)-1));
                else
                    Gps.vtgSpd(iEns,1)=NaN;
                end


            else
                Gps.vtgTrue(iEns,1)=NaN;
                Gps.vtgMagN(iEns,1)=NaN;
                Gps.vtgSpd(iEns,1)=NaN;
            end
            if (idataTypes+1)<=Hdr.nDataTypes(1)

fseek(fidWR,(Hdr.dataOffsets(1,idataTypes+1)+fileloc),'bof');
            else
                fseek(fidWR,fileloc+bytesPerEns-2,'bof');
            end

%%      Read GSA NMEA String
        %Just reads in the string, the rest can be sorted out later
           idataTypes=idataTypes+1;
            marker=ftell(fidWR);
            fseek(fidWR,(Hdr.dataOffsets(1,idataTypes)+fileloc+2),'bof');
            dummy3=char(fread(fidWR,60,'char'))';
            dummy=zeros(1,60);
            endstr=find(dummy3==char(42))+2;
            dummy(1,1:endstr)=dummy3(1:endstr);

            Nmea.gsa(iEns,:)=dummy(1,:);

            if (idataTypes+1)<=Hdr.nDataTypes(1)

fseek(fidWR,(Hdr.dataOffsets(1,idataTypes+1)+fileloc),'bof');
            else
                fseek(fidWR,fileloc+bytesPerEns-2,'bof');
            end
        otherwise
           %disp(ftell(fidWR));
    end;
```

```
    if idataTypes==Hdr.nDataTypes(1)
        Inst.resRDI=fread(fidWR,1,'uint16');
        checksum=fread(fidWR,1,'uint16');
    end;
end;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%
%%%%Calculate Water Velocities and Boat Velocities
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%
%Calculate Boat Velocities
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
BTVelShip = Bt.vel_mps';  % BTVelShip(:,1) made negative to make it port,
not starboard.
BTVelShip(:,1)=-BTVelShip(:,1);

Bt.velE_mps=-
BTVelShip(:,1).*cosd((Sensor.heading_deg+MagVar))+BTVelShip(:,2).*sind((Se
nsor.heading_deg+MagVar));
Bt.velN_mps=BTVelShip(:,2).*cosd((Sensor.heading_deg+MagVar))+BTVelShip(:,
1).*sind((Sensor.heading_deg+MagVar));

Bt.velE_mps=-Bt.velE_mps;       %Boat East Velocity
Bt.velN_mps=-Bt.velN_mps;       %Boat North Velocity
Bt.velUp_mps=BTVelShip(:,3);    %Boat Upward Velocity
Bt.velError_mps=BTVelShip(:,4); %Boat Velocity Error


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%
%Calculate Water Velocities
%insert east and north water velocities into a 2D array
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%
for p=1:nEnsembles
    for q=1:nBins
        watVelE(p,q)=Wt.vel_mps(1,p,q);
    end
end

for p=1:nEnsembles
    for q=1:nBins
        watVelN(p,q)=Wt.vel_mps(2,p,q);
        sensHeadingDeg(p,q)=Sensor.heading_deg(p);
    end
end
for p=1:nEnsembles
    for q=1:nBins
        watVelUp(p,q)=Wt.vel_mps(3,p,q);           %Water Vertical Velocity
```

```
    end
end


%These are water velocities for each bin in the ensemble
watVelE=-watVelE;

Wt.velE_mps=-watVelE.*cosd((sensHeadingDeg+MagVar))+
watVelN.*sind((sensHeadingDeg+MagVar));
Wt.velN_mps=watVelN.*cosd((sensHeadingDeg+MagVar))+watVelE.*sind((sensHead
ingDeg+MagVar));




%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%



%Convert time from format hhmmss to decimal seconds
    for k=1:length(Gps.ggaUTC)
        hours=floor(Gps.ggaUTC(k)/10000);
        minutes=floor(Gps.ggaUTC(k)/100)-hours*100;
        seconds=100*(Gps.ggaUTC(k)/100-floor(Gps.ggaUTC(k)/100));
        satTime_sec(k)=hours*3600+minutes*60+seconds;
    end

    satTime_sec=interpolateNaN(satTime_sec);
    antAltitude=interpolateNaN(antAltitude);

%Calculate water surface (boat) vertical velocity from GPS data
    for k=2:length(Gps.ggaUTC)
       delt(k-1)= satTime_sec(k)-satTime_sec(k-1);
       deltAlt(k-1)= antAltitude(k)-antAltitude(k-1);
       vertVel(k-1)=deltAlt(k-1)/ delt(k-1);
    end
    vertVel=vertVel';
    ggaensemble=Sensor.num;
    [x,y,z]=fileparts(fullName);

% Calculate modified (boat velocities subtracted) water vertical
velocities in each bim in the ensemble
for p=1:nEnsembles-1
    for q=1:nBins
        watVelUpMod(p,q)=watVelUp(p,q)-vertVel(p);          %Vertical
Water Velocity
    end
end
```

```
%should automatically save antAltitude and watVelUpMod, but we're doing
%this by hand after the run

%Add another row to watVelUpMod to have nEnsembles rows
watVelUpMod(end+1,:)=watVelUpMod(end,:);

%Write to Excel
    excelFlag=0; %set excel flag to 1 if you want to write to file, (0 if
not)
    if excelFlag
        excelname=y;

        titles={'Ens. #','ModWatVelUp m/s'};
        xlswrite(excelname,titles,'Sheet1','A1');
        xlswrite(excelname,[ggaensemble(1:end-
1),watVelUpMod],'Sheet1','A2');
    end
fclose('all');
```

B.2 THE MATLAB CODE FRASER_ALLFILES_FIRSTBIT_2

```
%This program processes ADCP transect data to assess the 3D structure of a
%spatial survey
%
%first read the binary data from a processed data file (processed in
WinRiver) into matlab using Rich Pavlowicz's code

filestoprocess={'Fraser2006_queenscalamity_June24_006r.000'}


boatref=input('specify boat velocity reference: 1 for bt, 2 for gps, 3 for
bt with gps, 4 for gps with bt, or 5 for kalman ');


m=1;

MagVar=18.083;

while m<=length(filestoprocess)
    fullName=filestoprocess{m}
    readADCPbinary_shorter
    dummyforsave=['save outdata' num2str(m+4)]
    eval(dummyforsave)
    save cycle filestoprocess m MagVar boatref
    clear
    load cycle
    m=m+1
end

%load outdata1
```

B.3 THE MATLAB CODE FRASER2006_ALLFLES_3

```
%This program processes ADCP transect data to assess the 3D structure of a
%spatial survey
%
%first read the binary data using the matlab binary extactor from USGS.
%The file used to do this was PicanocApril262006_allfiles_firstbit

%%first bit already run for binary stripper
filestoprocessafter={'Fraser2006_queenscalamity_June24_001r.000'
    'Fraser2006_queenscalamity_June24_002r.000'
    'Fraser2006_queenscalamity_June24_003r.000'
    'Fraser2006_queenscalamity_June24_004r.000'
    'Fraser2006_queenscalamity_June24_006r.000'}


 %   'Fraser2006_queenscalamity_June24_005r.000' %this file had problems
with the binary
 %   extractor, due to missing GPS data.


%MagVar=input('enter the magnetic declination (18.067 for
Fraser2006)');%actually, MagVar not required now, as binary stripper
%converts all to ENU coordinates
filename=input('enter filename in single quotes (eg ''Fraser2006'') ');
adcpdepth=input('enter the depth of ADCP transducers below water surface
(0.36 m for Fraser2006) ')
maxboatvel=input('enter the maximum possible boat velocity (5 m/s for
Fraser2006) (this will filter gps vel) ');
windowsize=input('enter an odd number of ensembles to moving average for
shear and depth estimates (probably about 11) ');
binstoaverage=input('enter the number of singleping ensembles to boxcar
average for plotting vectors (probably 11) ');

%make the structure arrays that I need
outdata.longitude=[];
outdata.latitude=[];
outdata.nav_east_vel=[];
outdata.nav_north_vel=[];
outdata.nav_vert_vel=[];
outdata.bt_east_vel=[];
outdata.bt_north_vel=[];
outdata.number=[];
outdata.time=[];
outdata.east_vel=[];
outdata.north_vel=[];
outdata.vert_vel=[];
```

```
outdata.bt_range=[];
outdata.pitch=[];
outdata.roll=[];
outdata.heading=[];
outdata.elev=[];
outdata.utme=[];
outdata.utmn=[];

%
m=1;

assumeddepthstop=[];

while m<=length(filestoprocessafter)

    dummyforload=['load outdata' num2str(m)]
    eval(dummyforload)

if m==1
    disttobin1=Cfg.distBin1_cm/100;
    numberdepthcells=Cfg.wn;
    binsize=Cfg.ws_cm/100;
    depths(1)=disttobin1+adcpdepth;
    k=2;
    while k<=numberdepthcells
        depths(k)=depths(1)+binsize*k;
        k=k+1;
    end

    cellsize=Cfg.ws_cm/100;%in m

    assumeddepth=depths(1)-cellsize;
    p=1;
    while assumeddepth>0
    assumeddepthstop=[assumeddepthstop; assumeddepth];
    p=p+1;
    assumeddepth=depths(1)-cellsize*p;
    end
    assumeddepthstop=flipud(assumeddepthstop);
    assumeddepthstop=assumeddepthstop';

end


    %calculate long lat in decimal degrees
    %make longitude negative for western hemisphere
    Gps.ggaLon=-Gps.ggaLon;

for k=1:length(Gps.ggaLon(:))
    if Gps.ggaLon(k)>0
        longdecdeg(k)=floor(Gps.ggaLon(k)/100)+(Gps.ggaLon(k)-
floor(Gps.ggaLon(k)/100)*100)/60;%negative for West hemisphere
```

```
        latdecdeg(k)=floor(Gps.ggaLat(k)/100)+(Gps.ggaLat(k)-
floor(Gps.ggaLat(k)/100)*100)/60;
    else
        longdecdeg(k)=ceil(Gps.ggaLon(k)/100)+(Gps.ggaLon(k)-
ceil(Gps.ggaLon(k)/100)*100)/60;%negative for West hemisphere
        latdecdeg(k)=floor(Gps.ggaLat(k)/100)+(Gps.ggaLat(k)-
floor(Gps.ggaLat(k)/100)*100)/60;
    end
end

%convert long and lat to utm
[utme,utmn]=findutmshort(longdecdeg,latdecdeg);

%elev=Gps.alt_m;
clear elev
dummyelev=['load antAltitude' num2str(m)]
eval(dummyelev)
elev=antAltitude;

elev=antAltitude;%this is from Shalini's extraction of GGA NMEA line
%correct GPS for base coordinate
if m==1 | m==2 | m==3%survey of June 24
    utme=utme+1.498;%see basecoordinates.xls
    utmn=utmn+(-2.756);
    elev=elev+(-1.120);
elseif m==4 | m==5%survey of June 25
    utme=utme+0.317;%see basecoordinates.xls
    utmn=utmn+(-3.361);
    elev=elev+(1.442);
end



%calculate average depth from four beams
for k=1:length(Bt.depth_m)
bt_range(k)=nanmean(Bt.depth_m(:,k))+adcpdepth;
end

%load shalini values for this file
%outdata.vert_vel=Shalini's values for this file
dummyup=['load watVelUpMod' num2str(m)]
eval(dummyup)
Wt.vel_mps(:,:,3)=watVelUpMod';

%load shalini values for vertical boat velocity - this should load a
%variable named vertVel

%save boatVelUpMod# vertVel

dummyup=['load vertVel' num2str(m)]
eval(dummyup)
vertVel=vertVel;
```

```
%concatenate the structure arrays that I need
outdata.longitude=cat(2,outdata.longitude,longdecdeg);
outdata.latitude=cat(2,outdata.latitude,latdecdeg);
outdata.nav_east_vel=cat(1,outdata.nav_east_vel,Gps.ggaVelE_mps);%note,
using RTK gga vel for boat velocity - this is m/s, should check rest of
code
outdata.nav_north_vel=cat(1,outdata.nav_north_vel,Gps.ggaVelN_mps);
outdata.bt_east_vel=cat(1,outdata.bt_east_vel,Bt.velE_mps);
outdata.bt_north_vel=cat(1,outdata.bt_north_vel,Bt.velN_mps);
outdata.number=cat(1,outdata.number,Sensor.num);
outdata.time=cat(1,outdata.time,Sensor.time);
outdata.east_vel=cat(2,outdata.east_vel,Wt.velE_mps);%assumes data
collected in ENU
outdata.north_vel=cat(2,outdata.north_vel,Wt.velN_mps);
outdata.vert_vel=cat(2,outdata.vert_vel,Wt.vel_mps(:,:,3));
outdata.bt_range=cat(2,outdata.bt_range,bt_range);%in m - check
outdata.pitch=cat(1,outdata.pitch,Sensor.pitch_deg);%in degrees - check
this too
outdata.roll=cat(1,outdata.roll,Sensor.roll_deg);
outdata.heading=cat(1,outdata.heading,Sensor.heading_deg);
outdata.elev=cat(1,outdata.elev,elev);
outdata.utme=cat(1,outdata.utme,utme);
outdata.utmn=cat(1,outdata.utmn,utmn);
outdata.nav_vert_vel=cat(1,outdata.nav_vert_vel,vertVel);%vertical boat
velocity from gps

save outdatasave outdata m filestoprocessafter filename adcpdepth
maxboatvel depths cellsize assumeddepthstop binstoaverage windowsize
boatref
clear
load outdatasave

%repeat for other files
    m=m+1
end

%it appears that I collected in ship coordinates, not earth coordinates
%(both Picanoc and Fraser 2006 surveys).
%so, I need to correct for heading

%convert Sensor.time to continous seconds
hour=outdata.time(:,1);
minute=outdata.time(:,2);
second=outdata.time(:,3)+outdata.time(:,4)/100;
timeseconds=hour.*(60*60)+minute.*60+second;
for k=1:(length(timeseconds)-1)
    timestep(k)=timeseconds(k+1)-timeseconds(k);
end
timestep(k+1)=timestep(k);%fill in a timestep for last ensemble
```

```
utme=outdata.utme;
utmn=outdata.utmn;

averagedepth=meanignoringNaNandzeroswithdim(outdata.bt_range,2);


%eliminate obvious gps errors
for k=1:length(outdata.nav_east_vel)
    if abs(outdata.nav_east_vel(k))>maxboatvel |
abs(outdata.nav_north_vel(k))>maxboatvel
        outdata.nav_east_vel(k)=NaN;
        outdata.nav_north_vel(k)=NaN;
        outdata.east_vel(:,k)=NaN;%eliminate the water velocities that
were calculated with bad gps boat ref
        outdata.north_vel(:,k)=NaN;%eliminate the water velocities that
were calculated with bad gps boat ref
    end
end

%find bedspeed
bedspeede=outdata.nav_east_vel-(outdata.bt_east_vel); %bedspeed is
calculated by subtracting bt from gps, because bt is now speed of boat wrt
bed
bedspeedn=outdata.nav_north_vel-(outdata.bt_north_vel); %make bt vel neg,
so vel is vel of boat wrt bed
%find average, but have to remove bad points first
badindex=find(isnan(bedspeede));
if ~isempty(badindex)
   averagebedspeede=meanignoringNaN(bedspeede)
   averagebedspeedn=meanignoringNaN(bedspeedn)
else
   averagebedspeede=mean(bedspeede)
   averagebedspeedn=mean(bedspeedn)
end
%find standard deviation, removing bad points
if ~isempty(badindex)
   stdbedspeede=stdignoringNaN(bedspeede)
   stdbedspeedn=stdignoringNaN(bedspeedn)
else
   stdbedspeede=std(bedspeede)
   stdbedspeedn=std(bedspeedn)
end

figure(1), clf
plot(outdata.number,outdata.nav_east_vel,outdata.number,outdata.bt_east_ve
l)
title([filename ' bottom track and GPS east velocity'])
legend('gps','bt')
xlabel('ensemble')
ylabel('east boat velocity, m/s')

figure(2), clf
plot(outdata.number,outdata.nav_north_vel,outdata.number,outdata.bt_north_
```

```
vel)
title([filename ' bottom track and GPS north velocity'])
legend('gps','bt')
xlabel('ensemble')
ylabel('north boat velocity, m/s')

figure(3), clf
plot(bedspeede,bedspeedn, 'g+',averagebedspeede,averagebedspeedn,'r*')
title([filename ' Bed Velocity, from GPS - BT Boat
Velocity'],'FontSize',14)
xlabel('Bed Speed East, m/s','FontSize',14)
ylabel('Bed Speed North, m/s','FontSize',14)
axis equal

%plot the courses in bottom track and GPS
%need to get bottom track course

ebtwonan=interpolateNaN(outdata.bt_east_vel);
nbtwonan=interpolateNaN(outdata.bt_north_vel);
%find the trajectory with the estimated velocity series
delebt=(ebtwonan).*timestep';
delnbt=(nbtwonan).*timestep';
trajectoryebt(1)=0;
trajectorynbt(1)=0;
for k=1:length(outdata.bt_east_vel)
   trajectoryebt(k+1)=delebt(k)+trajectoryebt(k);
   trajectorynbt(k+1)=delnbt(k)+trajectorynbt(k);
end
%
%now, replace estimated points with NaN - make trajectory a different
name, so that time series analysis can
%be performed on entire estimated set
badebtindex=find(isnan(outdata.bt_east_vel));
if ~isempty(badebtindex)
   countbad=1;
   for k=1:length(trajectoryebt)
      if k == badebtindex(countbad)
         trajectoryebtnan(k)=NaN;
         trajectorynbtnan(k)=NaN;
         if countbad < length(badebtindex)
            countbad=countbad+1;
         end
      else
         trajectoryebtnan(k)=trajectoryebt(k);
         trajectorynbtnan(k)=trajectorynbt(k);
      end
   end
else
   trajectoryebtnan=trajectoryebt;
   trajectorynbtnan=trajectorynbt;
end %end of initial check that there are some bad bt points
%
%
%
```

```
figure(4), clf
plot(utme,utmn,'-+')
%legend('GPS trajectory','bt trajectory',0)
title([filename ' Boat trajectory GPS'],'FontSize',12)
xlabel('East, m','FontSize',12)
ylabel('North, m','FontSize',12)
axis equal
%make a similar plot with 0,0 as first location
utmeendzeroed(1)=0;
utmnendzeroed(1)=0;
for k=1:length(utme)
utmeendzeroed(k+1)=utme(k)-utme(1);
utmnendzeroed(k+1)=utmn(k)-utmn(1);
end
figure(5), clf
plot(utmeendzeroed,utmnendzeroed,'-+',trajectoryebt,trajectorynbt,'-d')
legend('GPS trajectory','bt trajectory',0)
title([filename ' Boat trajectory GPS and bt (with interpolation for
missing values)'],'FontSize',12)
xlabel('East, m','FontSize',12)
ylabel('North, m','FontSize',12)
axis equal
%plot with NaN values
figure(6), clf
plot(utmeendzeroed,utmnendzeroed,trajectoryebtnan,trajectorynbtnan,'r-')
legend('GPS trajectory','Bottom track trajectory',0)
title([filename ' Boat Trajectory by GPS and Bottom
tracking'],'FontSize',14)
xlabel('East, m','FontSize',14)
ylabel('North, m','FontSize',14)
axis equal

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%I'm taking my old code from PHD, so I need to rename variables to match
[numensemb,width]=size(outdata.east_vel');
vele=outdata.east_vel';
veln=outdata.north_vel';
vertvel=outdata.vert_vel';

depth=outdata.bt_range;%I think this is m
gpsprofilenumber=outdata.number;

velesingleping=vele;%also save a single ping record of velocities, with
goodve and goodvesingleping found in calcshearADCP
velnsingleping=veln;

%smooth the shear and phi values
%by smoothing using moving averages of the velocities
%try a moving average to smooth out doppler noise

nformovavg=(windowsize-1)/2;
%windowsize=2*nformovavg+1%this the number of profiles on either side of
```

```
each profile that will be included in moving average
%note: moving average will smooth out real variance too
veletemp(1:nformovavg,:)=vele(1:nformovavg,:);
velntemp(1:nformovavg,:)=veln(1:nformovavg,:);
for k=(nformovavg+1):(numensemb-nformovavg)
% depth(k)=meanignoringNaNandzeros(depth((k-
nformovavg):(k+nformovavg)));%I
% won't moving average the depth - the depth will remain local
     for m=1:width
      veletemp(k,m)=meanignoringNaNandzeros(vele((k-
nformovavg):(k+nformovavg),m));
      velntemp(k,m)=meanignoringNaNandzeros(veln((k-
nformovavg):(k+nformovavg),m));
   end
end
veletemp((numensemb-nformovavg+1):numensemb,:)=vele((numensemb-
nformovavg+1):numensemb,:);
velntemp((numensemb-nformovavg+1):numensemb,:)=veln((numensemb-
nformovavg+1):numensemb,:);
clear vele veln
vele=veletemp;
veln=velntemp;

figure(7), clf
plot(gpsprofilenumber,depth,'-+')
title([filename ' Depth estimate from bottom tracking'],'FontSize',12)
xlabel('profile number','FontSize',12)
ylabel('depth to bottom, m','FontSize',12)

calcshearADCP

figure(8),clf
plot(gpsprofilenumber,shear,'+')
title([filename ' shear from individual profile'],'FontSize',12)
xlabel('profile','FontSize',12)
ylabel('shear vel, m/s','FontSize',12)
figure(9),clf
plot(gpsprofilenumber,ks,'+')
title([filename ' ks from individual profile'],'FontSize',12)
xlabel('profile','FontSize',12)
ylabel('ks, m','FontSize',12)
figure(10),clf
plot(gpsprofilenumber,phi,'+')
title([filename ' phi from individual profile'],'FontSize',12)
xlabel('profile','FontSize',12)
ylabel('phi, rad','FontSize',12)

figure(11),clf;
% plot shear and depth vs profile number - use
hl1=plot(gpsprofilenumber,shear,'b-');
% assign current axis handle to variable for later reference if needed
ax1=gca;
% set properties of the axes
set(ax1,'YMinorTick','on', 'box','on','ycolor',get(hl1,'color'));
```

```
% add title to plot
htitle = title(['shear and depth for each profile']);
ylabel('Shear (solid)')
xlabel('profile')
% add 1st floating axis for the second parameter (salinity) plotted
[hl2,ax2,ax3] = floatAxisY(gpsprofilenumber,depth,'r:','Depth
(dotted)');%,[0 100 32 34]


figure(12),clf
plot(depth,shear,'+')
title([filename ' shear vs depth'],'FontSize',12)
xlabel('depth','FontSize',12)
ylabel('shear vel, m/s','FontSize',12)
[shearwoNaN,depthwoNaN]=removeNaNandInftwoinputs(shear,depth);%note: I
can't just remove NaN, as all profiles with bad depths produce Inf shear,
due to divide by zero
%regression(shearwoNaN,depthwoNaN)

figure(15),clf
plot(depth,ks,'+')
title([filename ' ks vs depth'],'FontSize',12)
xlabel('depth','FontSize',12)
ylabel('ks, m/s','FontSize',12)


%%write the depth matrix for surfer plotting
%[utmedplot, utmndplot, dplot]=removeNaNthreeinputs(utme, utmn, depth');
%depthforsurf=[utmedplot; utmndplot; dplot]';
%wk1write('depthforsurf',depthforsurf,0,0);
%%write the shear matrix for surfer plotting
%[utmeshplot, utmnshplot, shplot]=removeNaNthreeinputs(utme, utmn,
shear');
%shearforsurf=[utmeshplot; utmnshplot; shplot]';
%wk1write('shearforsurf',shearforsurf,0,0);


for k=1:length(shear)
   if ks(k)<5 & ks(k)>0.0001
      shearforplot(k)=shear(k);
      ksforplot(k)=ks(k);
      depthforplot(k)=depth(k);
      utmeforplot(k)=utme(k);
      utmnforplot(k)=utmn(k);
   else
      shearforplot(k)=NaN;
      ksforplot(k)=NaN;
      depthforplot(k)=depth(k);
      utmeforplot(k)=utme(k);
      utmnforplot(k)=utmn(k);
   end
end
```

```
%write the matrix for surfer plotting
[utmeplot, utmnplot, dplot, shearplot,
ksplot]=removeNaNfiveinputs(utmeforplot', utmnforplot', depthforplot',
shearforplot', ksforplot');
filtshearforsurf=[utmeplot; utmnplot; dplot; shearplot; ksplot]';
wk1write('filtshearforsurf',filtshearforsurf,0,0);

figure(16),clf
plot(dplot,shearplot,'+')
title([filename ' shear vs depth, 0.0001<ks<5'],'FontSize',12)
xlabel('depth, m','FontSize',12)
ylabel('shear vel, m/s','FontSize',12)
%regression(shearplot,dplot)
figure(20),clf
semilogy(dplot,ksplot,'+')
title([filename ' ks vs depth, 0.0001<ks<5'],'FontSize',12)
xlabel('depth, m','FontSize',12)
ylabel('ks, m','FontSize',12)

%meanprimaryvel=meanignoringNaNwithdim(primaryvel,2);
%figure(21),clf
%plot(meanprimaryvel,contourlny(61,:),'+-')
%title([filename ' mean primaryvel profile'],'FontSize',12)
%ylabel('ln depth (depth in m)','FontSize',12)
%xlabel('primaryvel, m/s','FontSize',12)
%
%[lengthgood,widthgood]=size(primaryvel);
%meanprimaryveleachprofile=meanignoringNaNandzeroswithdim(primaryvel,1);
%for k=1:lengthgood
%   for m=1:binsinprofile(k)
%
nondimprimaryvel(k,m)=primaryvel(k,m)./meanprimaryveleachprofile(k);
%      bindepths(k,m)=depth(k)-contourdepth(k,m);
%      nondimbindepths(k,m)=bindepths(k,m)./depth(k);
%   end
%end
%
%
%figure(22),clf
%semilogy(nondimprimaryvel,nondimbindepths,'+')
%title([filename ' non dim primaryvel profile'],'FontSize',12)
%ylabel('depthabovebed/totaldepth','FontSize',12)
%xlabel('primaryvel/meanprimaryvelinprofile','FontSize',12)
%figure(23),clf
%plot(nondimprimaryvel,log(nondimbindepths),'+')
%title([filename ' non dim primaryvel profile over first
hour'],'FontSize',12)
%ylabel('ln of depthabovebed/totaldepth','FontSize',12)
%xlabel('primaryvel/meanprimaryvelinprofile','FontSize',12)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
ADCPvectorplots

%write the matrices for surfer plotting of depth average vel
[utmeplotwater, utmnplotwater,averageeastvelplot,averagenorthvelplot
]=removeNaNfourinputs(utme,utmn,averageeastvel',averagenorthvel');
watervelforsurf=[utmeplotwater;
utmnplotwater;averageeastvelplot;averagenorthvelplot]';
wk1write('watervelforsurf',watervelforsurf,0,0);

%write the matrices for surfer plotting of bedspeed
[utmeplotbs, utmnplotbs,bedspeedeplot,bedspeednplot
]=removeNaNfourinputs(utme,utmn,bedspeede,bedspeedn);
bedspeedforsurf=[utmeplotbs;utmnplotbs;bedspeedeplot;bedspeednplot]';
wk1write('bedspeedforsurf',bedspeedforsurf,0,0);

%average flow direction
averagedirection=nanmean(phi)

extractfortecplot_3
```

B.4 THE MATLAB CODE

MEANIGNORINGNANANDZEROSWITHDIM

```
function [out] = meanignoringNaNandzeroswithdim(in,dim)

%This is function 'meanignoringNaN', that will determine the mean of an
array
%while ignoring NaN values or non-values.
%find average, but remove bad (NaN) points first


%takes the mean along the dimension DIM of input.
%
%   Example: If input = [0 1 2
%                        3 4 5]
%
%   then mean(input,1) is [1.5 2.5 3.5] and mean(input,2) is [1
%                                                              4]
%I'VE GOT THIS WRONG: DIM=1 is average of rows, and DIM=2 is average of
columns


if prod(size(in))==1, out = in; return, end

if nargin==1,
  dim = min(find(size(in)~=1));
  if isempty(dim), dim = 1; end
end


line=1;

while line <= size(in,dim)
   if dim==2
      numberrows=size(in,1);
      sumin=sum(in,1);
     if ~isempty(in(:,line))
        badindex=find(isnan(in(:,line)));
        count=1;
        sumall=0;
        countzeros=0;
        %find number of zeroes
        for k=1:size(in,1)
           if in(k,line)==0
              countzeros=countzeros+1;
```

```
        end
      end

      %find sum without bad points
        if length(badindex) > 0
           for k=1:numberrows
              if k~=badindex(count)
                 sumall=sumall+in(k,line);
              else
                 if count ~= length(badindex)
                    count=count+1;
                 end
              end
           end
              out(line)=sumall/(numberrows-length(badindex)-countzeros);
        else
              out(line) = sumin(line)./(numberrows-countzeros);

        end
   else
      out(line) = NaN;
   end
end
if dim==1
   numbercolumns=size(in,2);
   sumin=sum(in,2);
   if ~isempty(in(line,:)),
      badindex=find(isnan(in(line,:)));
      count=1;
      sumall=0;
      countzeros=0;
      %find number of zeroes
      for k=1:size(in,2)
         if in(line,k)==0
            countzeros=countzeros+1;
         end
      end

      %find sum without bad points
        if length(badindex) > 0
           for k=1:numbercolumns
              if k~=badindex(count)
                 sumall=sumall+in(line,k);
              else
                 if count ~= length(badindex)
                    count=count+1;
                 end
              end
           end
           numbercolumns=size(in,2);
           out(line)=sumall/(numbercolumns-length(badindex)-countzeros);
        else
           out(line) = sumin(line)./(numbercolumns-countzeros);
        end
```

```
    else
       out(line) = NaN;
    end
  end

 line=line+1;


end
```

B.5 THE MATLAB CODE MEANIGNORINGNAN

```
function [means] = meanignoringNaN(input)
%This is function 'meanignoringNaN', that will determine the mean of an
array
%while ignoring NaN values or non-values
%find average, but remove bad (NaN) points first
badindex=find(isnan(input));
count=1;
sums=0;
countzeros=0;
 %find sum without bad points
 if length(badindex) > 0
   for k=1:length(input)
      if k~=badindex(count)
         sums=sums+input(k);
      else
         if count ~= length(badindex)
            count=count+1;
         end
      end
   end
   means=sums/(length(input)-length(badindex));
else
   means=sum(input)/(length(input));
end
```

B.6 THE MATLAB CODE STDIGNORINGNAN

```
function [stdignoringNaN] = stdignoringNaN(input)
%This is function 'sdignoringNaN', that will determine the standard
deviation of an array
%while ignoring NaN values as non-values.  Note: this uses std, which
normalizes by N-1.
badindex=find(isnan(input));
count=1;
counter=1;
if length(badindex)>0
   for k=1:length(input)
      if k~=badindex(count)
         inputwithoutNaN(counter)=input(k);
         counter=counter+1;
      else
         if count ~= length(badindex)
            count=count+1;
         end
      end
   end
   stdignoringNaN=std(inputwithoutNaN);
else
   stdignoringNaN=std(input);
end
```

B.7 THE MATLAB CODE CALCSHEARADCP

```
baddepthindex=find(isnan(depth));

badcount=1;
for k=1:numensemb
   l=1;
   repeater=1;
   badcount=1;
   if length(baddepthindex)>0
      if k==baddepthindex(badcount)
            goodve(k,:)=NaN;
            goodvn(k,:)=NaN;
            goodvesingleping(k,:)=NaN;
            goodvnsingleping(k,:)=NaN;
            goodvertvelsingleping(k,:)=NaN;
            contourdepth(k,:)=NaN;
            contourybin(k,:)=NaN;
            contourlny(k,:)=NaN;
            binsinprofile(k)=NaN;
            badcount=badcount+1;
      else
            while (((depth(k)-0.06*depth(k)) - depths(l))>0 & (depth(k) -
depths(l))>cellsize)%only loop through for each profile for the bins that
are shallower than depth for profile (above 6%sidelobe and above bottom
cell)
          % while ((depth(k) - depths(l))>0 & (depth(k) -
depths(l))>cellsize)%only loop through for each profile for the bins that
are shallower than depth for profile (above bottom cell)
            contourdepth(k,l)=depths(l);
            contourybin(k,l)=(depth(k) - depths(l));
            contourlny(k,l)=log(contourybin(k,l));
            goodve(k,l)=vele(k,l); %in m/s already
            goodvn(k,l)=veln(k,l);
            goodvesingleping(k,l)=velesingleping(k,l); %in m/s already
            goodvnsingleping(k,l)=velnsingleping(k,l);
            goodvertvelsingleping(k,l)=vertvel(k,l);
            repeater=repeater+1;
            l=l+1;
         end
         binsinprofile(k)=repeater-1;
      end
   else
            while (((depth(k)-0.06*depth(k)) - depths(l))>0 & (depth(k) -
depths(l))>cellsize)%only loop through for each profile for the bins that
are shallower than depth for profile (above 6%sidelobe and above bottom
cell)
%           while ((depth(k) - depths(l))>0 & (depth(k) -
```

```
depths(l))>cellsize)%only loop through for each profile for the bins that
are shallower than depth for profile (above 10%sidelobe and above bottom
cell)
            contourdepth(k,l)=depths(l);
            contourybin(k,l)=(depth(k) - depths(l));
            contourlny(k,l)=log(contourybin(k,l));
            goodve(k,l)=vele(k,l);  %in m/s already
            goodvn(k,l)=veln(k,l);
            goodvesingleping(k,l)=velesingleping(k,l); %in m/s already
            goodvnsingleping(k,l)=velnsingleping(k,l);
            goodvertvelsingleping(k,l)=vertvel(k,l);
            repeater=repeater+1;
            l=l+1;
         end
         binsinprofile(k)=repeater-1;

   end
end
%[numensembgood,widthgood]=size(goodve);
%for k=1:numensembgood
%   for m=1:widthgood
%      if goodve(k,m)==0
%         goodve(k,m)=NaN;
%      end
%      if goodvn(k,m)==0
%         goodvn(k,m)=NaN;
%      end
%   end
%end

%find vel vector angle
[contourvelvecangle,contourvelvec]=cart2pol(goodve,goodvn);
for k=1:size(contourvelvecangle,1)%make angles range from 0 to 2pi rather
than pi to -pi
   for l=1:size(contourvelvecangle,2)
      if contourvelvecangle(k,l)<0
         contourvelvecangle(k,l)=2*pi+contourvelvecangle(k,l);
      end
   end
end
plotcontourvelvecangle=contourvelvecangle.*180/pi;
%averagevelvecangle=meanignoringNaNwithdim(contourvelvecangle,2).*180/pi;
%averagevelvec=meanignoringNaNwithdim(contourvelvec,2);
%pixelplot(contourvelvec, contourdepth, depth, gpsprofilenumber, filename,
'velvector')
%pixelplot(plotcontourvelvecangle, contourdepth, depth, gpsprofilenumber,
filename, 'velvectorangle')
a=size(goodvn);
for k=1:a(1)
   depthforpixel(k)=depth(k);
   gpsprofilepixel(k)=gpsprofilenumber(k);
end
%pixelplot(goodvn, contourdepth, depthforpixel, gpsprofilepixel, filename,
'velnorth')
```

```
%pixelplot(goodve, contourdepth, depthforpixel, gpsprofilepixel, filename,
'veleast')

%determine the primary flow direction for each vertical
%use the Nature paper by Bathurst, Thorne, Hey 1977, because it deals with
a single vertical, not a cross-section
%take only the averageve and averagevn that occur above the average depth
%actually, take only points where the entire bin is above average depth
%That's not quite correct - a bin is calculated using 2 cell sizes, with
the centre of the bin between the two cells
%(see Sontek Principles of Operation p5) therefore, can only get within
one cell size of the bed.
badcount=1;
[ldepth,wdepth]=size(contourdepth);
for k=1:numensemb
   if length(baddepthindex)>0
      if k==baddepthindex(badcount)
         phi(k)=NaN;
         if badcount<length(baddepthindex)
            badcount=badcount+1;
         end
      else
         m=1;
         while contourdepth(k,m)>0
            ve(m)=goodve(k,m);
            vn(m)=goodvn(k,m);
            d(m)=contourdepth(k,m);
            m=m+1;
            if m==wdepth
               break
            end
         end
      if m==1
          phi(k)=NaN;
      else

         %assume that velocity at surface is equal to velocity at first
bin
         topve=[];
         topvn=[];
         for ntop=1:length(assumeddepthstop);
             topve=[topve goodve(k,1)];
             topvn=[topvn goodvn(k,1)];
         end
         assumedve = [topve ve];
         assumedvn = [topvn vn];
         assumeddepths = [assumeddepthstop d];
         integralvn=trapz(assumeddepths,assumedvn);
         integralve=trapz(assumeddepths,assumedve);
         clear assumedve assumedvn assumeddepths ve vn d
         if integralvn < 0
            if integralve < 0
               phi(k)=atan(integralvn/integralve)+pi;
            else
```

```
                phi(k)=atan(integralvn/integralve)+2*pi;
            end
        else
            if integralve < 0
                phi(k)=pi+atan(integralvn/integralve);
            else
                phi(k)=atan(integralvn/integralve);
            end
        end
    end
    end
  else
            m=1;
            while contourdepth(k,m)>0
                ve(m)=goodve(k,m);
                vn(m)=goodvn(k,m);
                d(m)=contourdepth(k,m);
                m=m+1;
                if m==wdepth
                    break
                end
            end
      if m==1
            phi(k)=NaN;
      else
            %assume that velocity at surface is equal to velocity at first
bin
        topve=[];
        topvn=[];
        for ntop=1:length(assumeddepthstop);
            topve=[topve goodve(k,1)];
            topvn=[topvn goodvn(k,1)];
        end
        assumedve = [topve ve];
        assumedvn = [topvn vn];
        assumeddepths = [assumeddepthstop d];
            integralvn=trapz(assumeddepths,assumedvn);
            integralve=trapz(assumeddepths,assumedve);
            clear assumedve assumedvn assumeddepths ve vn d
            if integralvn < 0
                if integralve < 0
                    phi(k)=atan(integralvn/integralve)+pi;
                else
                    phi(k)=atan(integralvn/integralve)+2*pi;
                end
            else
                if integralve < 0
                    phi(k)=pi+atan(integralvn/integralve);
                else
                    phi(k)=atan(integralvn/integralve);
                end
            end
      end
   end
```

```
end
%I should also calculate a shear stress for each individual profile and
then average
%rotate all velocities by phi

for k=1:a(1)
   phiforpixel(k)=phi(k);
   for m=1:length(goodve(k,:))

primaryvel(k,m)=goodve(k,m).*cos(phiforpixel(k))+goodvn(k,m).*sin(phiforpi
xel(k));
   secondaryvel(k,m)=goodvn(k,m).*cos(phiforpixel(k))-
goodve(k,m).*sin(phiforpixel(k));
   end
end


%calculate average primary vel again, this time from the individual
primary vel - it may be different
%note, this doesn't mean much now, as the phi changes for every profile,
therefore the mean primary vel
%throughout the reach is not an average of one flow direction
%primaryvelaverageafter=meanignoringNaNandzeroswithdim(primaryvel,2);
%secondaryvelaverageafter=meanignoringNaNandzeroswithdim(secondaryvel,2);
%primaryvelstdafter=stdignoringNaNandzeroswithdim(primaryvel,2);
%secondaryvelstdafter=stdignoringNaNandzeroswithdim(secondaryvel,2);
%primaryvelminafter=min(primaryvel);
%primaryvelmaxafter=max(primaryvel);
%secondaryvelminafter=min(primaryvel);
%secondaryvelmaxafter=max(primaryvel);

%
%expectedstd=140*1454.2/(1500000*cellsize*(averaginginterval*9*length(good
ve))^0.5)

for k=1:numensemb
%find the shear stress for each profile
if binsinprofile(k)>2

[ufitaverage,ufitaverageerror]=polyfit(contourlny(k,1:binsinprofile(k)),pr
imaryvel(k,1:binsinprofile(k)),1);
       shear(k)=0.41*ufitaverage(1);
       ks(k)=30/exp(ufitaverage(2)/ufitaverage(1));
   else
      shear(k)=NaN;
      ks(k)=NaN;
   end
end
```

B.8 THE MATLAB CODE FLOATAXISY

```
function [hl1,ax2,ax3] = floatAxisY(varargin)
% floatAxisY  create floating x-axis for multi-parameter plot
%
%=======================================================================
% floatAxisY  Version 1.2 6-Mar-2000
%
% Usage:
%   [h1,ax2,ax3] = floatAxisY(varargin)
%
% Description:
%   This Matlab function creates a floating y-axis for mutli-parameter
%   plots with different units on the same figure. For example, in
oceanography,
%   it is common to plot temperature, salinity, and density versus depth.
%
%
% Input:
%   A minimum of two parameters is required. The first and second
parameters are
%   the x,y pairs to plot. The third parameter (optional) specifies the
linestyle
%   (defaults to 'k-' solid black). The fourth parameter (optional)
specifies the
%   y-axis label for the floating axis. The fifth parameter (optional)
specifies
%   the x and y limits for the axis(this should be of the form
%   [xlower xupper ylower yupper]).
%
% Output:
%   n/a
%
% Called by:
%   CTDplotY.m - script to demo floatAxis function
%
% Calls:
%   n/a
%
% Author:
%   Blair Greenan
%   Bedford Institute of Oceanography
%   18-May-1999
%   Matlab 5.2.1
%   greenanb@mar.dfo-mpo.gc.ca
%
%=======================================================================
%
```

```
% History
% Version 1.0 18-May-1999
% Version 1.1 31-May-1999
%    Added the ability to pass an array containing the x and y limits for
%    the axis.
% Version 1.2 6-Mar-2000
%    Added code to handle data with different x-limits. Previous versions
%    assumed all data had the same x-limits. Oops! Thanks to Jan Even
Nilsen
%    (even@gfi.uib.no) for pointing this out.


% strip the arguments passed in
if (nargin < 2)
   error('floatAxis requires a minimum of three parameters')
elseif (nargin == 2)
   x = varargin{1};
   y = varargin{2};
   % default lines style (solid black line)
   lstyle = 'k-';
elseif (nargin == 3)
   x = varargin{1};
   y = varargin{2};
   lstyle = varargin{3};
elseif (nargin == 4)
   x = varargin{1};
   y = varargin{2};
   lstyle = varargin{3};
   ylbl = varargin{4};
elseif (nargin == 5)
   x = varargin{1};
   y = varargin{2};
   lstyle = varargin{3};
   ylbl = varargin{4};
   limits = varargin{5};
else
   error('Too many arguments')
end

% get position of axes
allAxes = get(gcf,'Children');
ax1Pos = get(allAxes(length(allAxes)),'position');

% rescale and reposition all axes to handle additional axes
for ii = 1:length(allAxes)-1
   if (rem(ii,2)==0)
      % even ones in array of axes handles represent axes on which lines
are plotted
      set(allAxes(ii),'Position',[ax1Pos(1)+0.1 ax1Pos(2) ax1Pos(3)-0.1
ax1Pos(4)])
   else
      % odd ones in array of axes handles represent axes on which floating
```

```
x-axss exist
      axPos = get(allAxes(ii),'Position');
      set(allAxes(ii),'Position',[axPos(1)+0.1 axPos(2) axPos(3)
axPos(4)])
   end
end
% first axis a special case (doesn't fall into even/odd scenario of figure
children)
set(allAxes(length(allAxes)),'Position',[ax1Pos(1)+0.1 ax1Pos(2)
ax1Pos(3)-0.1 ax1Pos(4)])
xlimit1 = get(allAxes(length(allAxes)),'Xlim');

% get new position for plotting area of figure
ax1Pos = get(allAxes(length(allAxes)),'position');

% axis to which the floating axes will be referenced
ref_axis = allAxes(1);
refPosition = get(ref_axis,'position');

% overlay new axes on the existing one
ax2 = axes('Position',ax1Pos);
% plot data and return handle for the line
hl1 = plot(x,y,lstyle);
% make the new axes invisible, leaving only the line visible
set(ax2,'visible','off','xlim',xlimit1)

if (nargin < 5)
   % get the y limits for the
   ylimit = get(ax2,'YLim');
else
   set(ax2,'XLim',[limits(1) limits(2)],'YLim',[limits(3) limits(4)]);
end

% set the axis limit mode so that it does not change if the
% user resizes the figure window
set(ax2,'YLimMode','manual')

% set up another set of axes to act as floater
ax3 = axes('Position',[refPosition(1)-0.1 refPosition(2) 0.01
refPosition(4)]);
set(ax3,'box','off','ycolor','w','xticklabel',[],'xtick',[])
set(ax3,'YMinorTick','on','color','none','ycolor',get(hl1,'color'))
if (nargin < 5)
   set(ax3,'YLim',ylimit)
else
   set(ax3,'XLim',[limits(1) limits(2)],'YLim',[limits(3) limits(4)])
end

% label the axis
if (nargin > 3)
   ylabel(ylbl)
end
```

B.9 THE MATLAB CODE ADCPVECTORPLOTS

```
%run this after PicanocApril262006)_allfiles
%note that the previous file provides opportunity to average several
%ensembles together (to calculate shear).  This file does not use
%these moving averages, but the input binstoaverage provdes a boxcar
%average.  This reduces the number and smooths the vectors to be plotted.

%do depth average
averageeastvel=meanignoringNaNandzeroswithdim(goodvesingleping,1);
averagenorthvel=meanignoringNaNandzeroswithdim(goodvnsingleping,1);

figure(24), clf
quiver(utme,utmn,averageeastvel,averagenorthvel,6)
title([filename ' depth average water velocity'],'FontSize',14)
xlabel('Easting, m','FontSize',14)
ylabel('Northing, m','FontSize',14)

%need to average to see the arrows - try every five ensembles
averageutme=[];
averageutmn=[];
averageaverageeastvel=[];
averageaveragenorthvel=[];
averagetime=[];
%binstoaverage=10;%Note: the velocity profiles have already been moving
averaged when calculating shear
%this averaging will just reduce the number of vector arrows.  Note that
it
%may be better to do this averaging on raw vectors, but I would have to
%rewrite the code for PicanocApril26_allfiles

count=1;
k=1;
while k < (length(utme)-binstoaverage)
    averageutme(count)=meanignoringNaN(utme(k:k+binstoaverage-1));
    averageutmn(count)=meanignoringNaN(utmn(k:k+binstoaverage-1));

averagelong(count)=meanignoringNaN(outdata.longitude(k:k+binstoaverage-
1));
    averagelat(count)=meanignoringNaN(outdata.latitude(k:k+binstoaverage-
1));

averageaverageeastvel(count)=meanignoringNaN(averageeastvel(k:k+binstoaver
age-1));

averageaveragenorthvel(count)=meanignoringNaN(averagenorthvel(k:k+binstoav
erage-1));
```

```
    averagetime(count)=meanignoringNaN(outdata.time(k:(k+binstoaverage-
1)));

binaveragedepth(count)=meanignoringNaN(averagedepth(k:(k+binstoaverage-
1)));
    count=count+1;
    k=k+binstoaverage;
end

elow=min(utme)-100;
nlow=min(utmn)-100;
nhi=max(utmn)+100;
ehi=max(utme)+100;

vectorscaleeast=0.5/sqrt(2);%for a scale vector of 0.5 m/s
vectorscalenorth=0.5/sqrt(2);
utmescale=elow +50;
utmnscale=nlow + 50;
averageutmeplot=[averageutme utmescale];
averageutmnplot=[averageutmn utmnscale];
averageaverageeastvelplot=[averageaverageeastvel vectorscaleeast];
averageaveragenorthvelplot=[averageaveragenorthvel vectorscalenorth];
quivervecscale=0.5


figure(25), clf
quiver(averageutmeplot,averageutmnplot,averageaverageeastvelplot,averageav
eragenorthvelplot,quivervecscale,'k')
title([filename ' depth average water velocity , ' num2str(binstoaverage)
' ensembles boxcar average '],'FontSize',8)
xlabel('Easting, m','FontSize',14)
ylabel('Northing, m','FontSize',14)
axis equal
axis([elow, ehi, nlow, nhi])
```

B.10 THE MATLAB CODE REMOVENANFOURINPUTS

```
%this function removes NaN values from all four matrices, where if there
is NaN in one matrix, all four elements are removed
function [ebtwonan,secondwonan,thirdwonan,fourthwonan] =
removeNaNfourinputs(ebt,second,third,fourth)
badebtindex=find(isnan(ebt));
badsecondindex=find(isnan(second));
badthirdindex=find(isnan(third));
badfourthindex=find(isnan(fourth));
badebtindex=union(badebtindex,badsecondindex);
badebtindex=union(badebtindex,badthirdindex);
badebtindex=union(badebtindex,badfourthindex);

for k=1:length(ebt)
   btprofilecount(k)=k;
end
goodebtindex=setdiff(btprofilecount,badebtindex);
if ~isempty(badebtindex)
   countgood=1;
   countbad=1;
   for k=1:min([length(ebt) length(second) length(third) length(fourth)])
      if k~=badebtindex(countbad)
         ebtwonan(countgood)=ebt(k);
         secondwonan(countgood)=second(k);
         thirdwonan(countgood)=third(k);
         fourthwonan(countgood)=fourth(k);
         if countgood < length(goodebtindex)
            countgood=countgood+1;
         end
      else
         if countbad < length(badebtindex)
            countbad=countbad+1;
         end
      end
   end
else
   ebtwonan=ebt;
   secondwonan=second;
   thirdwonan=third;
   fourthwonan=fourth;
end
```

B.11 THE MATLAB CODE EXTRACTFORTECPLOT_3

```
%make data format for Tecplot



%correct water surface elevation for GPS rover antenna height
antheight=1.20%in m
outdata.elev=outdata.elev-antheight;

%get vertical velocity, but correct for boat motion
[length,width]=size(goodvesingleping)
sizevel=size(goodvesingleping);
numforutme=sizevel(2)
numutme=sizevel(1)

%correct for bad rtk gps data near the mountain - base correction must
have
%been shielded at this location
for s=1:numutme
if utme(s)>575373 & utme(s)<575598 & utmn(s)>5451528 & utmn(s)<5451728
    outdata.elev(s)=8.25;

goodvertvelsingleping(s,:)=goodvertvelsingleping(s,:)+outdata.nav_vert_vel
(s);%for locations with bad gps elevation, remove boat vertical velocity
correction from the vertical water velocities
end
end

%good water vel, with zeros still
vefortecplottemp=goodvesingleping';
vefortecplot=vefortecplottemp(:);
vnfortecplottemp=goodvnsingleping';
vnfortecplot=vnfortecplottemp(:);
vvertfortecplottemp=goodvertvelsingleping';
vvertfortecplot=vvertfortecplottemp(:);


%for k=1:(length-1)
%    boatvertvel(k)=(outdata.elev(k+1)-outdata.elev(k))/timestep(k);
%end
%boatvertvel=vertVel;


%make utme and utmn with enough points - this should be based on the width
%of the vel array,
```

```
utmefortecplottemp=[];
utmnfortecplottemp=[];
for k=1:sizevel(1)
utmefortecplottemp=cat(1,utmefortecplottemp,utme(k)*ones(numforutme,1));
utmnfortecplottemp=cat(1,utmnfortecplottemp,utmn(k)*ones(numforutme,1));
end
utmefortecplot=utmefortecplottemp;
utmnfortecplot=utmnfortecplottemp;


%utme utme utme utme utme utme utme utme utme utme ...
%    utme utme utme utme utme utme utme utme utme utme ...
%    utme utme utme utme utme utme utme utme utme utme ...
%    utme utme utme utme utme utme utme utme utme utme ...
%    utme utme utme utme utme utme utme utme utme utme ...
%    utme utme utme utme utme utme utme utme utme utme ...
%    utme utme utme]';
%utmefortecplot=utmefortecplottemp(:);

%utmnfortecplottemp=[utmn utmn utmn utmn utmn utmn utmn utmn utmn utmn ...
%    utmn utmn utmn utmn utmn utmn utmn utmn utmn utmn ...
%    utmn utmn utmn utmn utmn utmn utmn utmn utmn utmn ...
%    utmn utmn utmn utmn utmn utmn utmn utmn utmn utmn ...
%    utmn utmn utmn utmn utmn utmn utmn utmn utmn utmn ...
%    utmn utmn utmn utmn utmn utmn utmn utmn utmn utmn ...
%    utmn utmn utmn]';
%utmnfortecplot=utmnfortecplottemp(:);


%for now, assume depth represents the absolute elevation z.  I should use
%gps elevation, accounting for difference in base coordinates on each day.
depthfortecplottemp=contourdepth';
depthfortecplot=depthfortecplottemp(:);

for p=1:length
    for q=1:width
        elevfortecplottemp(p,q)=outdata.elev(p)-depths(q);
    end
end
elevfortecplottemp=elevfortecplottemp';
elevfortecplot=elevfortecplottemp(:);

bottomelev=outdata.elev-depth';%the elevation of bottom for each ensemble.


%create horizontal water surface by vertical transfer of elevations
P1east=min(utme)
P1ensemble=find(utme==P1east)
P1north=utmn(P1ensemble)
P2north=max(utmn)
P2ensemble=find(utmn==P2north)
P2east=utme(P2ensemble)
TotalDist=sqrt((P1east-P2east)^2+(P1north-P2north)^2)
```

```
slope=0.000357
%slope=(outdata.elev(P2ensemble)-outdata.elev(P1ensemble))/TotalDist %this
%doesn't work because the mineast point is a bad gps point with elevation
%of -0.2.  We should check that its easting and northing positions are
%reasonable.


%correct the water surface elevations and the bottom boundary elevations
for k=1:max(size(utme))
    %calculate distance to the point from the upstream and downstream ends
    distP1(k)=sqrt((P1east-utme(k))^2+(P1north-utmn(k))^2);
    distP2(k)=sqrt((P2east-utme(k))^2+(P2north-utmn(k))^2);
    %calculate the new water surface nad the new boundary elevation for a
flat water surface
    if distP1(k)>3000
        elevnew(k)=outdata.elev(k)-slope*distP1(k);
        bottomelevnew(k)=bottomelev(k)-slope*distP1(k);
    else
        elevnew(k)=outdata.elev(k)-(TotalDist-distP2(k))*slope;
        bottomelevnew(k)=bottomelev(k)-(TotalDist-distP2(k))*slope;
    end
end


%now, correct the water velocity elevations
for k=1:max(size(utmefortecplot))
    %calculate distance to the point from the upstream and downstream ends
    distP1fortecplot(k)=sqrt((P1east-utmefortecplot(k))^2+(P1north-
utmnfortecplot(k))^2);
    distP2fortecplot(k)=sqrt((P2east-utmefortecplot(k))^2+(P2north-
utmnfortecplot(k))^2);
    %calculate the new velocity point elevation for a flat water surface
    if distP1fortecplot(k)>3000
        elevnewfortecplot(k)=elevfortecplot(k)-slope*distP1fortecplot(k);
    else
        elevnewfortecplot(k)=elevfortecplot(k)-(TotalDist-
distP2fortecplot(k))*slope;
    end
end


%remove bad data and zeros - memory trouble, so do have at a time
halflength=fix(max(size(utmefortecplot))/2);
[utmefortecplotsmall1, utmnfortecplotsmall1, vefortecplotsmall1,
vnfortecplotsmall1,vvertfortecplotsmall1,
elevfortecplotsmall1]=removeNaNandzerossixinputs(utmefortecplot(1:halfleng
th), utmnfortecplot(1:halflength), vefortecplot(1:halflength),
vnfortecplot(1:halflength),vvertfortecplot(1:halflength),
elevnewfortecplot(1:halflength));
[utmefortecplotsmall2, utmnfortecplotsmall2, vefortecplotsmall2,
vnfortecplotsmall2,vvertfortecplotsmall2,
elevfortecplotsmall2]=removeNaNandzerossixinputs(utmefortecplot((halflengt
h+1):end), utmnfortecplot((halflength+1):end),
```

```
vefortecplot((halflength+1):end),
vnfortecplot((halflength+1):end),vvertfortecplot((halflength+1):end),
elevnewfortecplot((halflength+1):end));
utmefortecplotsmall=[utmefortecplotsmall1 utmefortecplotsmall2];
utmnfortecplotsmall=[utmnfortecplotsmall1 utmnfortecplotsmall2];
vefortecplotsmall=[vefortecplotsmall1 vefortecplotsmall2];
vnfortecplotsmall=[vnfortecplotsmall1 vnfortecplotsmall2];
vvertfortecplotsmall=[vvertfortecplotsmall1 vvertfortecplotsmall2];
elevfortecplotsmall=[elevfortecplotsmall1 elevfortecplotsmall2];

tecplotmatrixexport=[utmefortecplotsmall; utmnfortecplotsmall;
elevfortecplotsmall; vefortecplotsmall; vnfortecplotsmall;
vvertfortecplotsmall]';

tecplotboundaryexport=[utme utmn bottomelevnew'];

tecplotwatersurfexport=[utme utmn elevnew'];

%velocity data
save tecplotmatrixexport tecplotmatrixexport
save tecplotmatrixexportascii.txt tecplotmatrixexport -ascii

%bottom boundary
save tecplotboundaryexport tecplotboundaryexport
save tecplotboundaryexportascii.txt tecplotboundaryexport -ascii

%water surface
save tecplotwatersurfexport tecplotwatersurfexport
save tecplotwatersurfexportascii.txt tecplotwatersurfexport -ascii
wk1write('tecplotwatersurfexportascii', tecplotwatersurfexport)


%make a data file that includes zero vel at boundary for each element.
zeroscolumn=zeros(length,1);
boundarywithzeros=[utme utmn bottomelevnew' zeroscolumn zeroscolumn
zeroscolumn];
wk1write('tecplotboundaryexportascii', tecplotboundaryexport)

%water vel and a zero vel at boundary (for better interpolation with
Tecplot).
alldatawithboundary=[tecplotmatrixexport; boundarywithzeros];
save alldatawithboundary alldatawithboundary
save alldatawithboundaryascii.txt alldatawithboundary -ascii

save allfraserdatanew
```

C CREATING INTERPOLATED GRID IN SURFER

The 600 m average reach width and the 45° reach angle were the governing parameter of modelled variograms for the water surface elevation and the reach boundary. The following pages show the experimental variograms of the water surface elevation and the reach boundary, as well as the modelled variograms in the average streamwise and cross-streamwise directions. To view the variograms in the other directions refer to the appendix CD.

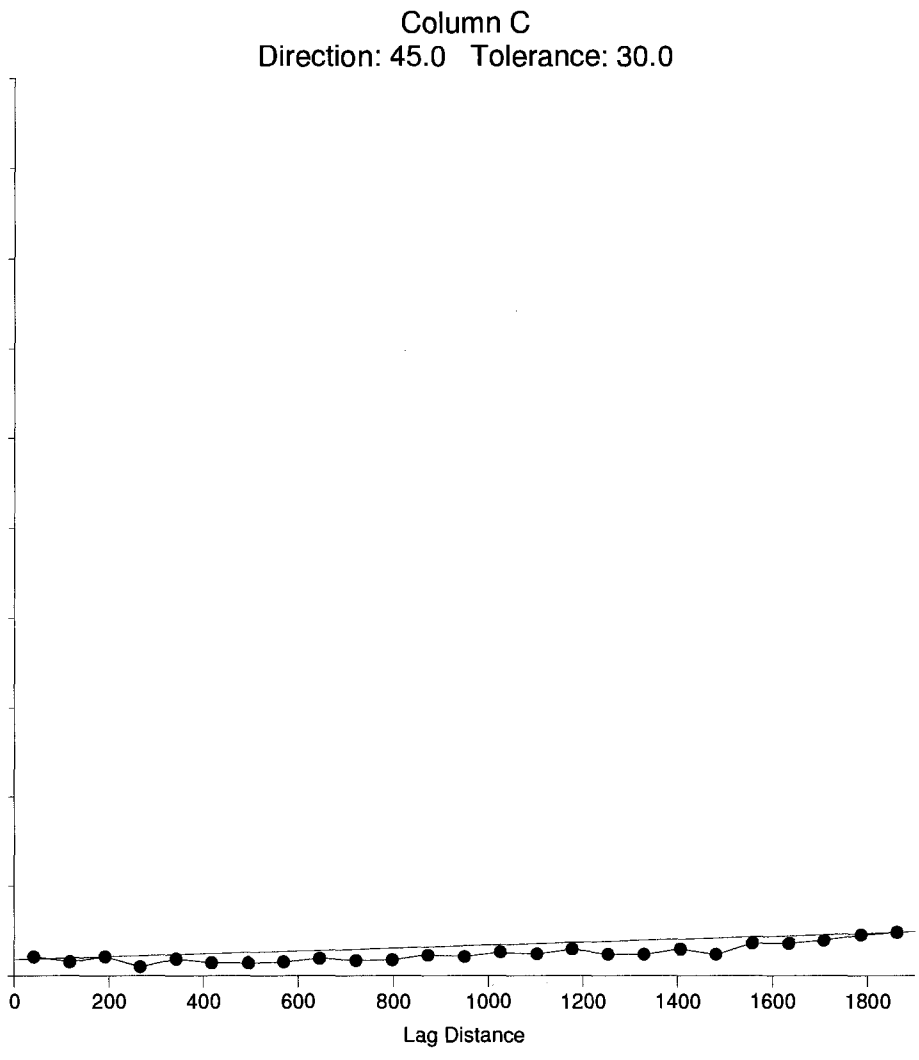

Figure C-7-1 The experimental and modelled water surface elevation variogram in streamwise direction. It is noticeable that in the streamwise direction, and after transferring the data to create a horizontal water surface elevations, the streamwise variogram is almost horizontal and linear.

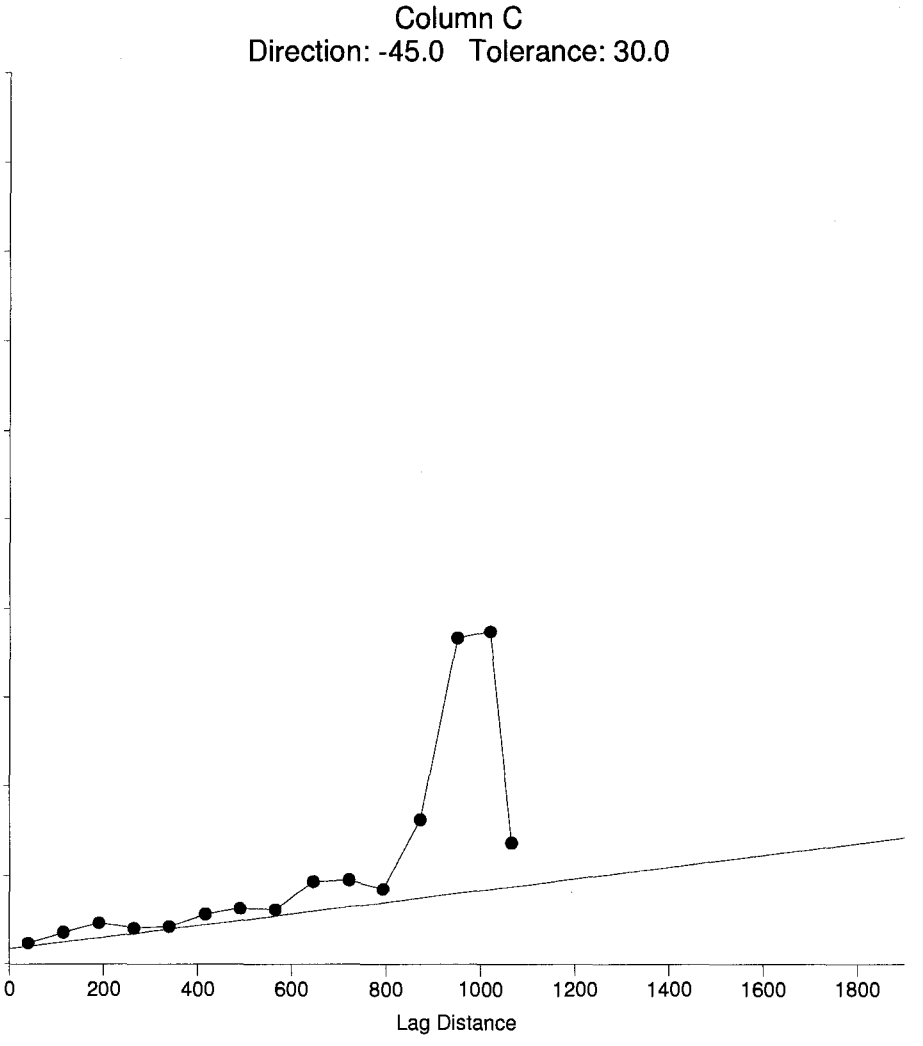

Figure C-7-2 The experimental and modelled water surface elevation variogram in cross-streamwise direction. It is noticeable that up to the lag distance of 600 m, which is the average reach width, the experimental variogram is linear.

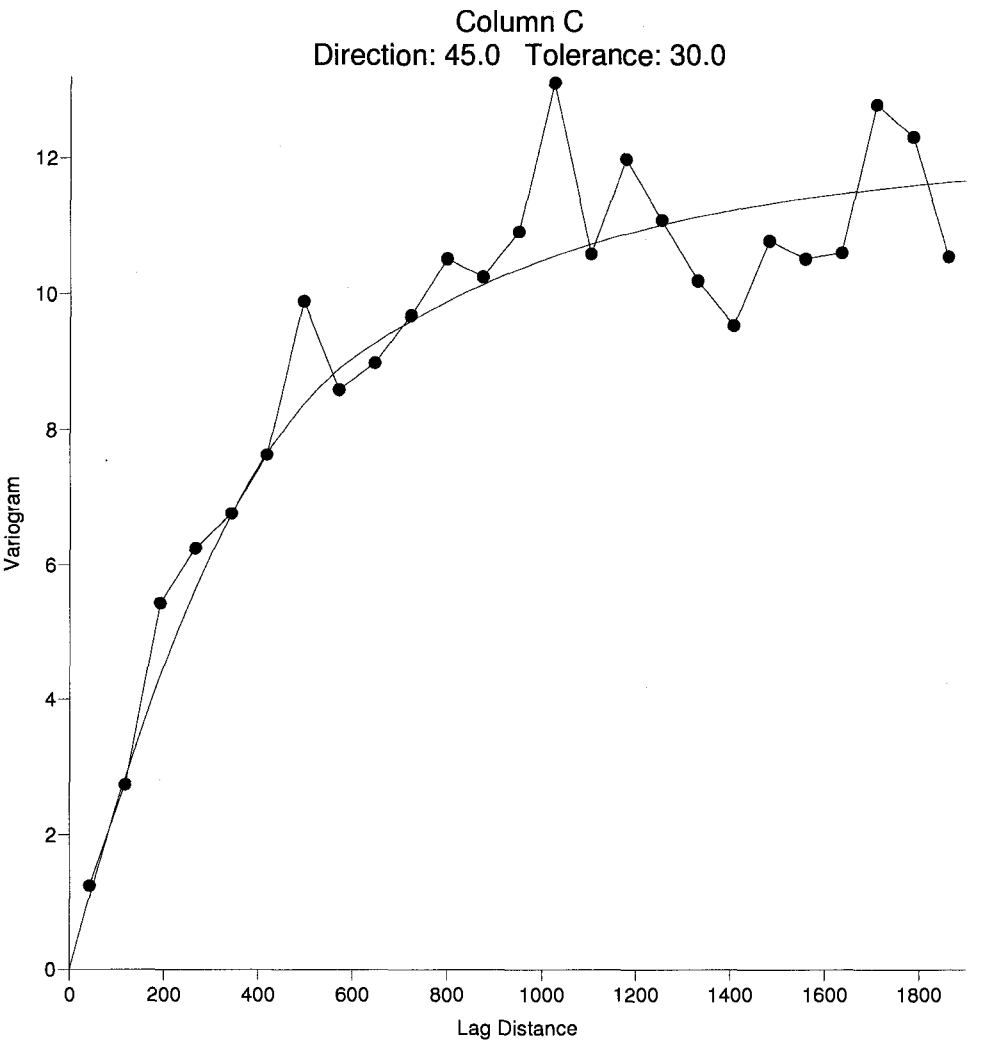

Figure C-7-3 The experimental and modelled reach boundary elevation variogram in streamwise direction.

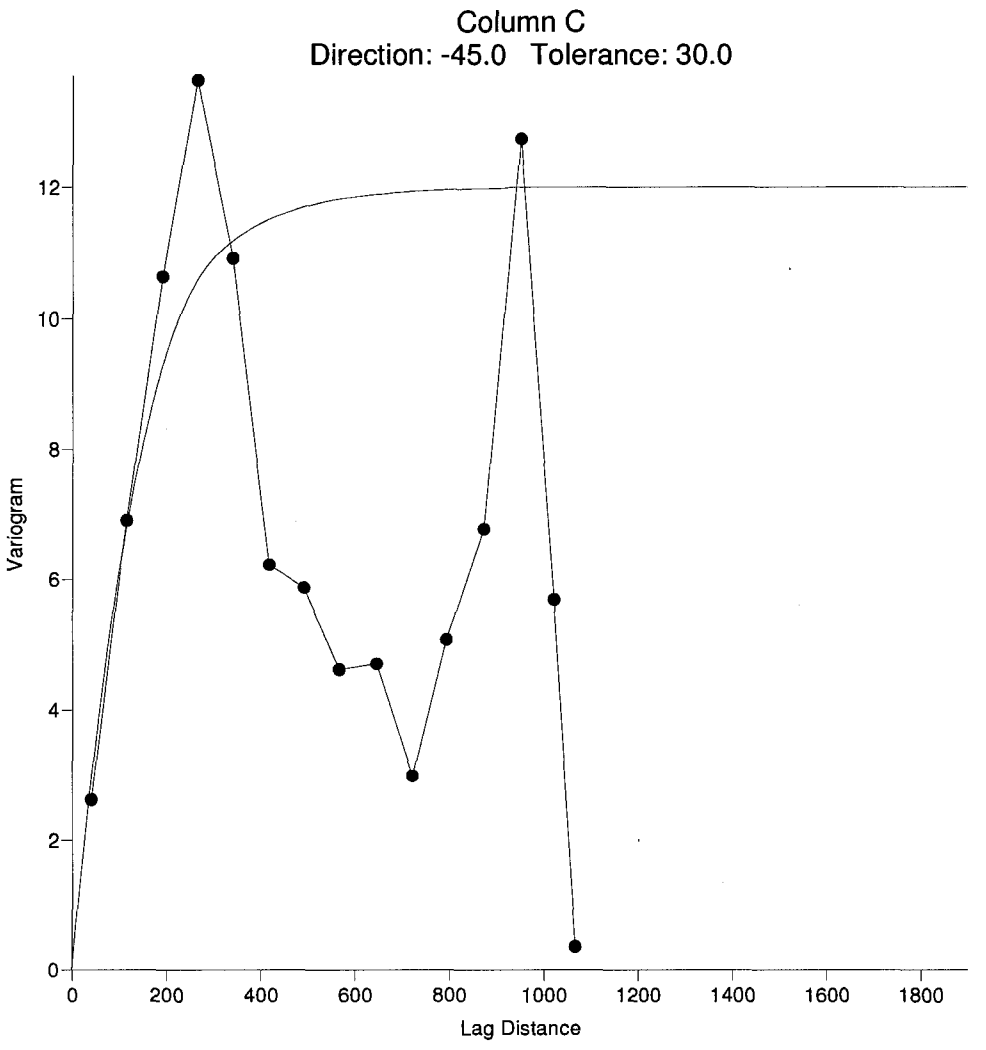

Figure C-7-4 The The experimental and modelled reach boundary elevation variogram in cross-streamwise direction. It is noticeable that up to the lag distance of 300 m, which is the half of the reach width, the experimental variogram increase linearly and then up to lag distance of 600 m, which is the reach width, it decreases linearly.

D THE MATLAB OUTPUT FILES

D.1 THE MATLAB OUTPUT FILE TECPLOTMATRIXEXPORTASCIIWITHERROR

The following shows the first few pages of the files tecplotmatrixexportasciiwitherror. To view the whole files refer to the appendix CD.

utme	utmn	elev	velocity-e	velocity-n	velocity-ver	error-vel
574105.92	5450478.90	6.097	-1.01015170	-1.60256490	-0.24300000	-0.104
574105.92	5450478.90	6.097	-1.01015170	-1.60256490	-0.24300000	-0.104
574105.92	5450478.90	5.597	-0.98712680	-1.79786690	-0.29100000	0.206
574105.92	5450478.90	5.347	-0.66434079	-1.56953780	-0.20500000	-0.058
574105.92	5450478.90	5.097	-0.65732661	-1.62929400	-0.10200000	-0.088
574105.92	5450478.90	4.847	-0.61049299	-1.27866080	-0.23800000	-0.049
574105.92	5450478.90	4.597	-0.58175265	-1.07527450	-0.29500000	-0.298
574105.92	5450478.90	4.347	-0.29271305	-1.15966120	-0.25700000	-0.047
574105.92	5450478.90	4.097	-0.21384518	-1.31175940	-0.20000000	-0.442
574105.85	5450479.40	6.107	-1.27903970	-1.73000080	0.08033333	0.162
574105.85	5450479.40	5.607	-1.00320650	-1.84949150	-0.09766667	0.058
574105.85	5450479.40	5.357	-0.71435359	-1.89444710	-0.17666667	-0.025
574105.85	5450479.40	5.107	-1.00830010	-2.04186630	-0.18166667	-0.139
574105.85	5450479.40	4.857	-1.01344090	-2.11469190	-0.10566667	0.154
574105.85	5450479.40	4.607	-1.15868070	-2.12970530	-0.06966667	-0.080
574105.85	5450479.40	4.357	-1.06934320	-1.49729420	-0.04866667	-0.403
574105.85	5450479.40	4.107	-0.85925605	-1.61911950	-0.03266667	-0.014
574105.77	5450480.00	6.093	-1.20850720	-2.09530570	-0.22800000	-0.012
574105.77	5450480.00	5.593	-1.34212640	-1.87660520	-0.21600000	-0.019
574105.77	5450480.00	5.343	-1.39516090	-1.56489330	-0.09300000	-0.095
574105.77	5450480.00	5.093	-1.20748030	-1.59310660	-0.08900000	-0.105
574105.77	5450480.00	4.843	-1.15564770	-1.93365530	-0.12800000	0.338
574105.77	5450480.00	4.593	-1.06573450	-1.90803070	-0.01500000	0.304
574105.77	5450480.00	4.343	-0.91720805	-1.82149230	-0.10000000	0.495
574105.77	5450480.00	4.093	-0.49440528	-1.42693200	-0.25500000	0.099
574105.72	5450480.60	6.111	-0.89324640	-1.85470570	-0.07433333	-0.114
574105.72	5450480.60	5.611	-1.01962410	-1.88517980	-0.15733333	-0.149
574105.72	5450480.60	5.361	-1.26181480	-1.85316910	-0.12433333	-0.089
574105.72	5450480.60	5.111	-1.22681360	-1.71434210	-0.15133333	-0.212
574105.72	5450480.60	4.861	-1.13686290	-1.75535280	-0.11333333	-0.425
574105.72	5450480.60	4.611	-0.92209560	-1.81559400	-0.05233333	-0.685
574105.72	5450480.60	4.361	-0.56194364	-1.38175700	-0.01533333	-0.857
574105.72	5450480.60	4.111	-0.62799112	-1.09085710	-0.03133333	-0.436

574105.69	5450481.30	6.119	-0.59658849	-1.48106260	-0.06200000	0.056
574105.69	5450481.30	5.619	-1.26548930	-1.21016650	-0.02600000	0.112
574105.69	5450481.30	5.369	-1.56214380	-1.31043160	0.03500000	0.295
574105.69	5450481.30	5.119	-0.77130674	-1.46366920	-0.04700000	0.168
574105.69	5450481.30	4.869	-0.98756600	-1.30811870	0.04600000	-0.120
574105.69	5450481.30	4.619	-0.77879771	-1.62141860	-0.03700000	0.271
574105.69	5450481.30	4.369	-0.92761108	-1.44723890	0.03000000	-0.203
574105.69	5450481.30	4.119	-0.70547059	-1.24532400	-0.01000000	-0.324
574105.70	5450481.90	6.122	-1.38267270	-2.12211600	-0.15866667	0.150
574105.70	5450481.90	5.622	-1.14937620	-1.98178520	-0.06266667	0.243
574105.70	5450481.90	5.372	-1.24285190	-2.37944870	-0.19266667	0.255
574105.70	5450481.90	5.122	-1.11545040	-2.06130270	-0.18366667	-0.298
574105.70	5450481.90	4.872	-1.00669450	-1.93172600	-0.20066667	0.197
574105.70	5450481.90	4.622	-0.69822371	-1.77393560	-0.18366667	0.292
574105.70	5450481.90	4.372	-0.48392116	-2.06409320	-0.13666667	-0.043
574105.70	5450481.90	4.122	-0.13822304	-1.88903660	-0.13266667	-0.118
574105.73	5450482.60	6.119	-1.23553720	-1.95854580	-0.03200000	0.084
574105.73	5450482.60	5.619	-1.07607380	-2.13242050	-0.07300000	0.067
574105.73	5450482.60	5.369	-0.59788208	-1.74393920	-0.02100000	0.001
574105.73	5450482.60	5.119	-0.90053595	-2.28317350	-0.06900000	0.561
574105.73	5450482.60	4.869	-0.80085439	-2.17605870	-0.03900000	0.492
574105.73	5450482.60	4.619	-0.82867305	-2.31747700	-0.02400000	0.833
574105.73	5450482.60	4.369	-0.73712389	-1.69998790	0.05000000	0.074
574105.73	5450482.60	4.119	-0.49534591	-1.77785580	-0.04400000	-0.200
574105.76	5450483.20	6.145	-0.57085230	-1.66928640	-0.00400000	-0.139
574105.76	5450483.20	5.645	-0.84339053	-1.55510700	0.00400000	-0.027
574105.76	5450483.20	5.395	-0.67190443	-1.33353870	-0.05900000	-0.396
574105.76	5450483.20	5.145	-0.94434110	-1.59478050	0.00200000	0.092
574105.76	5450483.20	4.895	-1.42240480	-1.85876560	-0.00900000	-0.095
574105.76	5450483.20	4.645	-0.97932349	-1.75104380	-0.03200000	-0.130
574105.76	5450483.20	4.395	-0.35298775	-1.94444640	-0.06900000	0.531
574105.82	5450483.80	6.134	-0.87422487	-1.42178120	0.13966667	-0.119
574105.82	5450483.80	5.634	-0.56948952	-1.91307880	0.17566667	0.480
574105.82	5450483.80	5.384	-0.72954794	-2.37654220	0.17366667	0.154
574105.82	5450483.80	5.134	-1.17294620	-2.12170300	0.30766667	0.362
574105.82	5450483.80	4.884	-0.70531219	-1.60868890	0.12066667	0.509
574105.82	5450483.80	4.634	-0.82137793	-1.48368590	0.09466667	-0.073
574105.82	5450483.80	4.384	-1.13472220	-1.87580890	0.17266667	0.019
574105.90	5450484.50	6.127	-1.55974890	-1.95806370	0.05200000	0.068
574105.90	5450484.50	5.627	-1.34619680	-1.41642220	0.05600000	0.437
574105.90	5450484.50	5.377	-0.92288774	-1.25832600	0.06100000	-0.289
574105.90	5450484.50	5.127	-0.83587394	-1.43776570	-0.18800000	-0.525
574105.90	5450484.50	4.877	-1.38510640	-1.60868000	-0.03700000	-0.184
574105.90	5450484.50	4.627	-1.10715560	-1.64889650	0.04600000	-0.055
574105.90	5450484.50	4.377	-1.11124480	-1.29955250	0.17000000	0.182

574105.98	5450485.10	6.126	-1.37055540	-2.13149910	-0.20633333	-0.394
574105.98	5450485.10	5.626	-1.19162560	-1.85227670	-0.15433333	-0.558
574105.98	5450485.10	5.376	-1.09572180	-1.70711650	-0.32633333	-0.160
574105.98	5450485.10	5.126	-1.16368260	-1.72476120	-0.20733333	0.034
574105.98	5450485.10	4.876	-0.94270649	-1.63876960	-0.10833333	-0.387
574105.98	5450485.10	4.626	-0.84196195	-1.35929050	-0.00833333	-0.176
574105.98	5450485.10	4.376	-0.72857247	-1.47077070	-0.06033333	-0.503
574106.15	5450485.70	6.101	-0.76440280	-1.94214780	-0.11200000	-0.207
574106.15	5450485.70	5.601	-0.93829375	-1.82423710	-0.11100000	0.114
574106.15	5450485.70	5.351	-0.91702707	-1.49729400	0.05700000	-0.260
574106.15	5450485.70	5.101	-0.52599577	-1.72972580	-0.06800000	-0.019
574106.15	5450485.70	4.851	-1.26438800	-1.82264980	0.03500000	-0.181
574106.15	5450485.70	4.601	-0.80284864	-1.25100100	-0.06300000	-0.034
574106.15	5450485.70	4.351	-0.35013497	-1.24763560	-0.04400000	-0.157
574106.27	5450486.20	6.173	-1.22147240	-2.03061510	0.06400000	-0.166
574106.27	5450486.20	5.673	-1.18310390	-1.94313450	0.04100000	0.111
574106.27	5450486.20	5.423	-0.97371555	-2.04028660	0.03500000	0.527
574106.27	5450486.20	5.173	-0.58890459	-2.03485260	0.01100000	0.394
574106.27	5450486.20	4.923	-0.81876554	-1.68243810	0.02300000	0.265
574106.27	5450486.20	4.673	-0.72990239	-1.72964340	-0.01900000	0.538
574106.27	5450486.20	4.423	-0.69847827	-1.78583530	0.06300000	-0.452
574106.43	5450486.70	6.145	-1.38569300	-1.72265160	0.16633333	-0.114
574106.43	5450486.70	5.645	-1.09586670	-1.71748350	0.12033333	-0.191
574106.43	5450486.70	5.395	-1.44023320	-2.21719210	0.13033333	0.028
574106.43	5450486.70	5.145	-1.48353660	-1.96111050	0.02433333	0.052
574106.43	5450486.70	4.895	-1.08603360	-1.73893840	-0.02966667	0.227
574106.43	5450486.70	4.645	-1.16455300	-1.52164890	-0.02366667	-0.016
574106.43	5450486.70	4.395	-1.24415490	-1.56214000	0.00333333	0.571
574106.59	5450487.20	6.113	-1.15409090	-1.91433980	0.18100000	0.224
574106.59	5450487.20	5.613	-1.38460440	-1.73123170	0.15400000	-0.501
574106.59	5450487.20	5.363	-1.27724650	-1.75571220	0.08600000	-0.043
574106.59	5450487.20	5.113	-1.42265620	-1.69549580	0.02100000	-0.033
574106.59	5450487.20	4.863	-1.08835670	-1.49567640	0.09900000	0.284
574106.59	5450487.20	4.613	-0.76097005	-1.93269700	0.14700000	-0.022
574106.59	5450487.20	4.363	-1.13113250	-1.22739440	0.22600000	0.131
574106.67	5450487.70	6.128	-1.31075010	-1.71138760	0.06000000	-0.042
574106.67	5450487.70	5.628	-1.61675760	-1.59178640	-0.01000000	0.162
574106.67	5450487.70	5.378	-1.03456110	-2.11880090	-0.09300000	0.168
574106.67	5450487.70	5.128	-0.91800596	-1.50548950	-0.02400000	-0.258
574106.67	5450487.70	4.878	-0.90164577	-1.01452990	-0.03100000	-0.209
574106.67	5450487.70	4.628	-0.54605004	-1.45510300	-0.02700000	-0.292
574106.67	5450487.70	4.378	-1.16109870	-1.77129950	-0.08700000	-0.018
574106.79	5450488.10	6.107	-1.41297990	-2.19482960	0.01900000	0.390
574106.79	5450488.10	5.607	-1.29553100	-1.89453180	0.00400000	0.310
574106.79	5450488.10	5.357	-1.29341850	-2.09092050	0.04100000	-0.359

574106.79	5450488.10	5.107	-1.16207780	-1.77935250	0.02200000	-0.236
574106.79	5450488.10	4.857	-0.94423289	-1.95275260	-0.01200000	-0.058
574106.79	5450488.10	4.607	-0.91476108	-1.72786060	-0.02200000	-0.142
574106.79	5450488.10	4.357	-0.59899265	-1.21771910	-0.02500000	-0.286
574106.90	5450488.50	6.096	-0.91923448	-2.18964940	-0.00100000	-0.036
574106.90	5450488.50	5.596	-0.58006066	-1.58941100	-0.03900000	-0.056
574106.90	5450488.50	5.346	-0.73719328	-1.64112500	0.00500000	0.168
574106.90	5450488.50	5.096	-1.21167860	-2.07259190	0.03200000	0.170
574106.90	5450488.50	4.846	-1.14463660	-1.63229500	0.02700000	-0.154
574106.90	5450488.50	4.596	-0.94753296	-2.03857590	-0.02900000	-0.207
574107.02	5450489.00	6.105	-1.62876290	-1.92245600	-0.25200000	-0.129
574107.02	5450489.00	5.605	-0.85641002	-2.13457620	-0.19600000	0.298
574107.02	5450489.00	5.355	-1.04323170	-2.24533560	-0.20700000	-0.368
574107.02	5450489.00	5.105	-1.14524120	-2.57277880	-0.31200000	-0.157
574107.02	5450489.00	4.855	-1.16521790	-2.16130960	-0.29300000	0.324
574107.02	5450489.00	4.605	-0.69714433	-1.64299970	-0.22200000	0.446
574107.11	5450489.40	6.121	-1.54131040	-2.03466660	0.14300000	-0.089
574107.11	5450489.40	5.621	-1.54428350	-1.52886410	-0.00600000	0.041
574107.11	5450489.40	5.371	-1.70917640	-2.05786160	0.08000000	0.329
574107.11	5450489.40	5.121	-1.60028790	-2.10833240	0.11900000	0.268
574107.11	5450489.40	4.871	-1.98857340	-2.03375080	0.16200000	0.274
574107.11	5450489.40	4.621	-1.43666570	-2.20639740	0.11300000	0.237
574107.24	5450489.80	6.099	-0.96099819	-1.45090340	-0.23633333	0.448
574107.24	5450489.80	5.599	-1.36197400	-2.07781200	-0.13633333	-0.219
574107.24	5450489.80	5.349	-1.07508560	-2.13933500	-0.14733333	-0.139

D.2 THE MATLAB OUTPUT FILE TECPLOTBOUNDARYEXPORTASCII

The following shows the first few pages of the files tecplotboundaryexportascii. To view the whole files refer to the appendix CD.

utme	utmn	bottom elev
574096.02	5450452.00	5.81270540
574096.22	5450452.50	5.75304480
574096.41	5450453.20	5.76636590
574096.50	5450453.70	5.82223220
574096.64	5450454.30	5.74008610
574096.78	5450454.90	4.17592280
574096.89	5450455.50	5.84678130
574097.06	5450456.10	5.83161370
574097.44	5450457.50	5.85072920
574097.62	5450458.20	5.85254590
574097.78	5450458.80	4.12536700
574097.84	5450459.70	5.92619320
574097.95	5450460.30	5.81853600
574097.97	5450461.00	5.85290330
574098.05	5450461.70	5.83374120
574098.22	5450462.20	5.78709860
574098.40	5450462.70	5.13145050
574098.76	5450463.20	6.68723460
574099.11	5450463.70	6.63304160
574099.45	5450464.20	6.67483100
574099.77	5450464.70	6.67564320
574100.17	5450465.30	6.68741600
574100.51	5450465.80	5.83571940
574100.88	5450466.20	5.85451770
574101.25	5450466.80	4.23880380
574102.21	5450468.00	5.83528600
574102.57	5450468.50	5.90907050
574103.02	5450469.20	5.87631030
574103.39	5450469.80	5.78909210
574103.64	5450470.30	4.23190560
574103.97	5450471.00	5.84317860

574104.24	5450471.60	5.82498950
574104.46	5450472.10	5.88131860
574104.76	5450472.80	5.87659780
574105.04	5450473.40	5.84340810
574105.31	5450473.90	4.24522630
574105.59	5450474.40	5.89404260
574105.87	5450475.10	5.90482310
574106.05	5450475.70	5.87565050
574106.16	5450476.30	5.88300440
574106.18	5450477.00	5.89387450
574106.15	5450477.40	4.29379770
574106.01	5450478.40	5.12765360
574105.92	5450478.90	3.63958620
574105.85	5450479.40	3.66201240
574105.77	5450480.00	3.69541190
574105.72	5450480.60	3.75581870
574105.69	5450481.30	3.79619180
574105.70	5450481.90	3.82157640
574105.73	5450482.60	3.85443070
574105.76	5450483.20	3.91781150
574105.82	5450483.80	3.89668360
574105.90	5450484.50	3.93452830
574105.98	5450485.10	4.00089620
574106.15	5450485.70	3.95072410
574106.27	5450486.20	4.10259170
574106.43	5450486.70	4.00794630
574106.59	5450487.20	4.02329840
574106.67	5450487.70	4.09319330
574106.79	5450488.10	4.08456620
574106.90	5450488.50	4.11846160
574107.02	5450489.00	4.13483760
574107.11	5450489.40	4.16323740
574107.24	5450489.80	4.16612280
574107.40	5450490.20	4.19849290
574107.57	5450490.60	4.19337540
574107.75	5450490.90	4.23525240
574108.01	5450491.40	4.22108400
574108.24	5450491.80	4.16942710
574108.48	5450492.40	4.19924450
574108.66	5450492.90	4.24759990
574108.84	5450493.50	4.27342950
574108.93	5450494.10	4.29479500
574108.99	5450494.60	4.28867370
574108.96	5450495.20	4.34906570
574108.87	5450495.80	4.40398510

574108.84	5450496.40	4.40188750
574108.78	5450496.90	4.52581300
574108.83	5450497.50	4.37518720
574108.87	5450498.00	4.52207750
574108.91	5450498.60	4.61894890
574109.00	5450499.20	4.53929760
574109.03	5450500.00	4.66615570
574109.11	5450500.70	4.68249600
574109.20	5450501.40	4.75733110
574109.35	5450502.20	4.77464170
574109.52	5450503.00	4.74543400
574109.66	5450503.80	4.79772910
574109.74	5450504.50	4.83307530
574109.82	5450505.30	4.86740740
574109.88	5450506.00	4.97226170
574109.98	5450506.60	4.90961210
574110.08	5450507.20	4.94797610
574110.19	5450507.70	4.97284180
574110.26	5450508.20	4.98173240
574110.35	5450508.70	5.00009740
574110.42	5450509.40	5.01445870
574110.43	5450510.00	5.07833290
574110.44	5450510.60	5.09771670
574110.43	5450511.20	5.14960500
574110.44	5450511.90	5.21398530
574110.52	5450512.50	5.17534360
574110.58	5450513.00	5.25322750
574110.69	5450513.60	5.24907620
574110.82	5450514.30	5.26691260
574110.95	5450515.00	5.30724710
574111.15	5450515.60	5.28306610
574111.34	5450516.10	5.33141100
574111.59	5450516.80	5.38171670
574111.87	5450517.40	5.33350780
574112.13	5450518.10	5.31230590
574112.33	5450518.70	5.36463040
574112.50	5450519.30	5.35546410
574112.65	5450519.80	5.39282810
574112.84	5450520.30	5.41316670
574113.08	5450520.80	5.41799530
574113.35	5450521.40	5.45731670
574113.63	5450521.80	5.39665020
574113.94	5450522.40	5.42545180
574114.16	5450523.00	5.47327400
574114.29	5450523.50	5.42162660

574114.31 5450524.10 5.49452080

D.3 THE MATLAB OUTPUT FILE TECPLOTWATERSURFEXPORTASCII

The following shows the first few pages of the files tecplotwatersurfexportascii. To view the whole files refer to the appendix CD.

utme	utmn	surf elev
574096.02	5450452.00	7.0502054
574096.22	5450452.50	7.0030448
574096.41	5450453.20	6.9938659
574096.50	5450453.70	7.0497322
574096.64	5450454.30	6.9775861
574096.78	5450454.90	6.9934228
574096.89	5450455.50	7.0392813
574097.06	5450456.10	7.0241137
574097.44	5450457.50	7.0207292
574097.62	5450458.20	7.0225459
574097.78	5450458.80	7.0153670
574097.84	5450459.70	7.0961932
574097.95	5450460.30	7.0110360
574097.97	5450461.00	7.0229033
574098.05	5450461.70	7.0037412
574098.22	5450462.20	6.9345986
574098.40	5450462.70	7.0264505
574098.76	5450463.20	7.0472346
574099.11	5450463.70	6.9930416
574099.45	5450464.20	7.0348310
574099.77	5450464.70	7.0356432
574100.17	5450465.30	7.0474160
574100.51	5450465.80	7.0082194
574100.88	5450466.20	7.0270177
574101.25	5450466.80	7.0538038
574102.21	5450468.00	7.0052860
574102.57	5450468.50	7.0790705
574103.02	5450469.20	7.0788103
574103.39	5450469.80	6.9915921
574103.64	5450470.30	7.0244056
574103.97	5450471.00	7.0231786

574104.24	5450471.60	6.9949895
574104.46	5450472.10	7.0388186
574104.76	5450472.80	7.0465978
574105.04	5450473.40	7.0234081
574105.31	5450473.90	7.0602263
574105.59	5450474.40	7.0640426
574105.87	5450475.10	7.0748231
574106.05	5450475.70	7.0456505
574106.16	5450476.30	7.0405044
574106.18	5450477.00	7.0513745
574106.15	5450477.40	7.0212977
574106.01	5450478.40	6.9851536
574105.92	5450478.90	7.0070862
574105.85	5450479.40	7.0170124
574105.77	5450480.00	7.0029119
574105.72	5450480.60	7.0208187
574105.69	5450481.30	7.0286918
574105.70	5450481.90	7.0315764
574105.73	5450482.60	7.0294307
574105.76	5450483.20	7.0553115
574105.82	5450483.80	7.0441836
574105.90	5450484.50	7.0370283
574105.98	5450485.10	7.0358962
574106.15	5450485.70	7.0107241
574106.27	5450486.20	7.0825917
574106.43	5450486.70	7.0554463
574106.59	5450487.20	7.0232984
574106.67	5450487.70	7.0381933
574106.79	5450488.10	7.0170662
574106.90	5450488.50	7.0059616
574107.02	5450489.00	7.0148376
574107.11	5450489.40	7.0307374
574107.24	5450489.80	7.0086228
574107.40	5450490.20	6.9984929
574107.57	5450490.60	6.9933754
574107.75	5450490.90	7.0252524
574108.01	5450491.40	7.0410840
574108.24	5450491.80	7.0019271
574108.48	5450492.40	6.9867445
574108.66	5450492.90	7.0125999
574108.84	5450493.50	7.0284295
574108.93	5450494.10	7.0472950
574108.99	5450494.60	7.0211737
574108.96	5450495.20	7.0240657
574108.87	5450495.80	7.0439851

574108.84	5450496.40	7.0218875
574108.78	5450496.90	7.0758130
574108.83	5450497.50	6.9726872
574108.87	5450498.00	6.9895775
574108.91	5450498.60	7.0414489
574109.00	5450499.20	7.0042976
574109.03	5450500.00	7.0311557
574109.11	5450500.70	7.0349960
574109.20	5450501.40	7.0548311
574109.35	5450502.20	7.0846417
574109.52	5450503.00	7.0654340
574109.66	5450503.80	7.0652291
574109.74	5450504.50	7.0530753
574109.82	5450505.30	7.0349074
574109.88	5450506.00	7.0597617
574109.98	5450506.60	7.0396121
574110.08	5450507.20	7.0354761
574110.19	5450507.70	7.0353418
574110.26	5450508.20	7.0342324
574110.35	5450508.70	7.0300974
574110.42	5450509.40	7.0069587
574110.43	5450510.00	7.0158329
574110.44	5450510.60	7.0027167
574110.43	5450511.20	7.0096050
574110.44	5450511.90	7.0514853
574110.52	5450512.50	7.0253436
574110.58	5450513.00	7.0432275
574110.69	5450513.60	7.0440762
574110.82	5450514.30	7.0369126
574110.95	5450515.00	7.0447471
574111.15	5450515.60	7.0205661
574111.34	5450516.10	7.0214110
574111.59	5450516.80	7.0642167
574111.87	5450517.40	7.0660078
574112.13	5450518.10	7.0348059
574112.33	5450518.70	7.0646304
574112.50	5450519.30	7.0454641
574112.65	5450519.80	7.0303281
574112.84	5450520.30	7.0281667
574113.08	5450520.80	7.0229953
574113.35	5450521.40	7.0598167
574113.63	5450521.80	7.0466502
574113.94	5450522.40	7.0504518
574114.16	5450523.00	7.0432740
574114.29	5450523.50	7.0041266

574114.31 5450524.10 7.0420208