

# WetSed User Manual

---

Prepared by Jessica Lovering  
& Peter Adams

March 9, 2012



## Contents

<b>1</b>	<b>Introduction and Background</b>	<b>1</b>
1.1	Model Overview and General Architecture . . . . .	1
1.2	Model Inputs . . . . .	2
1.2.1	Bathymetric Data . . . . .	2
1.2.2	Offshore Wave Climate . . . . .	2
1.3	Wave Transformation Modeling . . . . .	3
1.4	Longshore Sediment Transport Modeling . . . . .	3
1.4.1	Choosing the 5-meter Isobath . . . . .	4
1.4.2	Decimating Along the 5-meter Isobath . . . . .	4
1.4.3	Computing Coastal Trends . . . . .	4
1.4.4	Retrieving and Interpolating SWAN Output . . . . .	5
1.4.5	Calculating Angle of Incidence . . . . .	5
1.4.6	Calculating Wave Energy Flux . . . . .	6
1.4.7	Calculating Divergence of Drift . . . . .	6
1.5	Model Limitations . . . . .	7
1.6	References . . . . .	9
<b>2</b>	<b>Function Descriptions</b>	<b>10</b>
2.1	Model Flow . . . . .	10
2.2	wetsedExecuter.m . . . . .	11
2.2.1	Inputs and Outputs . . . . .	11
2.2.2	Steps . . . . .	11
2.3	swanControl.m . . . . .	12
2.3.1	Inputs and Outputs . . . . .	12
2.3.2	Steps . . . . .	12
2.4	swanPrjDetails.m . . . . .	13
2.4.1	Inputs and Outputs . . . . .	13
2.4.2	Steps . . . . .	13
2.5	swanDwwvmaker.m . . . . .	14
2.5.1	Inputs and Outputs . . . . .	14
2.5.2	Steps . . . . .	14
2.6	swanInputWriter.m . . . . .	15
2.6.1	Inputs and Outputs . . . . .	15
2.6.2	Steps . . . . .	15
2.7	swanCalcIJVals.m . . . . .	16
2.7.1	Inputs and Outputs . . . . .	16
2.7.2	Steps . . . . .	16
2.8	swanNestWriter.m . . . . .	17
2.8.1	Inputs and Outputs . . . . .	17

2.8.2	Steps	17
2.9	swanExecuter	18
2.9.1	Inputs and Outputs	18
2.9.2	Steps	18
2.10	cgemAnalyze.m	19
2.10.1	Inputs and Outputs	19
2.10.2	Steps	19
2.11	cgemDecimateIsobath.m	21
2.11.1	Inputs and Outputs	21
2.11.2	Steps	21
2.12	cgemComputeCsttrnd.m	22
2.12.1	Inputs and Outputs	22
2.12.2	Steps	22
2.13	cgemComputeWvEnergyFlx.m	23
2.13.1	Inputs and Outputs	23
2.13.2	Steps	23
2.14	cgemComputeDivDrift.m	24
2.14.1	Inputs and Outputs	24
2.14.2	Steps	24
<b>3</b>	<b>User Input Files to Modify</b>	<b>25</b>
<b>4</b>	<b>User Defined Input Files</b>	<b>26</b>
4.1	Main Grid Variables MAT File	26
4.2	Bathymetry Input Text Files	26
4.3	5m Isobath Location MAT File	26
<b>5</b>	<b>Definition of Variables</b>	<b>27</b>
<b>A</b>	<b>An Example using Simplified Geometric Coastal Shelves</b>	<b>A-1</b>
A.1	Introduction	A-1
A.2	Bathymetric Data	A-1
A.2.1	Planar Sloping Landscape	A-1
A.2.2	Sinusoidal Coastline Decaying with Depth	A-1
A.2.3	Exponentially Decaying Depth	A-2
A.2.4	Planar Slope with a Headland	A-2
A.2.5	Planar Slope with a Canyon	A-2
A.2.6	Planar Slope with a Hyperbolic Tangent Alongshore Trend	A-2
A.3	Wave Climate	A-4
A.4	Wave Transformation Results	A-4

A.5	Longshore Sediment Transport Model Output . . . . .	A-8
<b>B</b>	<b>An Example from the Southern California Bight</b>	<b>B-1</b>
B.1	Introduction . . . . .	B-1
B.2	Regional Setting . . . . .	B-1
B.3	Bathymetric Data . . . . .	B-2
B.4	Offshore Wave Climate . . . . .	B-2
B.5	Wave Transformation Results . . . . .	B-3
	B.5.1 Coarse Resolutions Wave Field Computation . . . . .	B-3
	B.5.2 Fine Resolutions Wave Field Computation . . . . .	B-4
B.6	Longshore Sediment Transport Model Output . . . . .	B-5
<b>C</b>	<b>Printed Matlab Codes</b>	<b>C-1</b>
C.1	wetsedExecuter.m . . . . .	C-1
C.2	wetsedInputVariables.m . . . . .	C-2
C.3	wetsedNestDetails.m . . . . .	C-3
C.4	swanControl.m . . . . .	C-4
C.5	swanPrjDetails.m . . . . .	C-7
C.6	swanDwwvmaker.m . . . . .	C-8
C.7	swanInputWriter.m . . . . .	C-9
C.8	cgemAnalyze.m . . . . .	C-12
C.9	cgemDecimateIsobath.m . . . . .	C-15
C.10	cgemComputeCsttrnd.m . . . . .	C-16
C.11	cgemComputeWvEnergyFlx.m . . . . .	C-19
C.12	cgemComputeDivDrift.m . . . . .	C-20
	<b>Index</b>	<b>i</b>

## List of Figures

1-1	Coastal Evolution Model architecture showing the relationship of the SWAN and CGEM modules . . . . .	1
A-1	The six simple geometric coastal shelves, each shown up to 200m water depth. . . . .	A-3
A-2	Sample of a simple SWAN input file . . . . .	A-4
A-3	Wave Heights and Directions computed by SWAN given the input conditions $H_s=2\text{m}$ , $T=10\text{sec}$ , and $\text{Dir}=135^\circ$ for six different bathymetries. . . . .	A-6
A-4	The percentage change in wave heights from a purely planar continental shelf for each of the five simple geometric bathymetries detailed in section A.2. . . . .	A-7
A-5	The step wise calculation of wave height and modeled wave direction along the 5m isobath, and the calculation of longshore sediment transport rates and divergence of drift for the planar sloping continental shelf bathymetry (described in section A.2.1) using the input conditions of $H_s=2\text{m}$ , $T=10\text{sec}$ , and $\text{Dir}=135^\circ$ . . . . .	A-9
A-6	The step wise calculation of wave height and modeled wave direction along the 5m isobath, and the calculation of longshore sediment transport rates and divergence of drift for the sinusoidal decaying continental shelf bathymetry (described in section A.2.2) using the input conditions of $H_s=2\text{m}$ , $T=10\text{sec}$ , and $\text{Dir}=135^\circ$ . . . . .	A-10
A-7	The step wise calculation of wave height and modeled wave direction along the 5m isobath, and the calculation of longshore sediment transport rates and divergence of drift for the continental shelf bathymetry with an exponentially decaying slope (described in section A.2.3) using the input conditions of $H_s=2\text{m}$ , $T=10\text{sec}$ , and $\text{Dir}=135^\circ$ . . . . .	A-11
A-8	The step wise calculation of wave height and modeled wave direction along the 5m isobath, and the calculation of longshore sediment transport rates and divergence of drift for the planar slope with a headland shaped continental shelf bathymetry (described in section A.2.4) using the input conditions of $H_s=2\text{m}$ , $T=10\text{sec}$ , and $\text{Dir}=135^\circ$ . . . . .	A-12
A-9	The step wise calculation of wave height and modeled wave direction along the 5m isobath, and the calculation of longshore sediment transport rates and divergence of drift for the planar slope with a canyon shaped continental shelf bathymetry (described in section A.2.5) using the input conditions of $H_s=2\text{m}$ , $T=10\text{sec}$ , and $\text{Dir}=135^\circ$ . . . . .	A-13

A-10	The step wise calculation of wave height and modeled wave direction along the 5m isobath, and the calculation of longshore sediment transport rates and divergence of drift for the planar slope with hyperbolic tangent shaped coastline shaped continental shelf bathymetry (described in section A.2.6) using the input conditions of $H_s=2m$ , $T=10sec$ , and $Dir=135^\circ$ . . . . .	A-14
A-11	The step wise calculation of wave height, longshore volumetric sediment transport rate and divergence of drift along the 5m isobath for the sinusoidal decaying continental shelf bathymetry (described in section A.2.2) using the input conditions of $H_s=2m$ , $T=10sec$ with varying modelled wave directions shown. . . . .	A-15
A-12	The step wise calculation of wave height, longshore volumetric sediment transport rate and divergence of drift along the 5m isobath for the planar slope with a headland shaped continental shelf bathymetry (described in section A.2.4) using the input conditions of $H_s=2m$ , $T=10sec$ with varying modelled wave directions shown. . . . .	A-16
A-13	The step wise calculation of wave height, longshore volumetric sediment transport rate and divergence of drift along the 5m isobath for the planar slope with a canyon shaped continental shelf bathymetry (described in section A.2.5) using the input conditions of $H_s=2m$ , $T=10sec$ with varying modelled wave directions shown. . . . .	A-17
A-14	The step wise calculation of wave height, longshore volumetric sediment transport rate and divergence of drift along the 5m isobath for the planar slope with hyperbolic tangent shaped coastline shaped continental shelf bathymetry (described in section A.2.6) using the input conditions of $H_s=2m$ , $T=10sec$ with varying modelled wave directions shown. . . . .	A-18
B-1	Map of the Southern California Bight showing the five nested regions used in this example for detailed modeling analysis . . . . .	B-1
B-2	Example SWAN runs showing wave height distribution over the entire Southern California Bight, for four separate sets of offshore wave conditions. . . . .	B-3
B-3	Example SWAN runs showing wave height distribution over Santa Barbara newst within the northern portion of the Southern California Bight, for four separate sets of offshore wave conditions. . . . .	B-4
B-4	Stepwise calculations of alongshore variability of modeled wave height, modeled wave direction, coast normal orientation, angle of incidence, and stress-flux factor for an example simulation in Santa Monica Bay. . . . .	B-5
B-5	Example calculation of longshore sediment transport rates and divergence of drift for Santa Monica Bay . . . . .	B-6

# 1 Introduction and Background

## 1.1 Model Overview and General Architecture

The model, **WetSed**(**W**ave **E**volution, **T**ransportation of **S**ediment, and **E**rosional **D**istribution), described in this manual was created to potential longshore sediment transport. It is generally organized into two major components: (1) SWAN, a freely available FORTRAN program, developed by the Delft Hydraulics Group (Booij et al., 1999), and (2) CGEM (Coastal Geomorphic Erosional Model), a series of MATLAB codes developed for this project. These two components compute the wave transformations and longshore sediment transport calculations, respectively, and interact by passing SWAN calculations to CGEM, as shown schematically in Figure 1-1. By proscribing a deep-water wave field (consisting of wave heights, periods, and directions) and digital bathymetry (topography of the ocean floor), SWAN calculates the patterns of refraction, diffraction, and redistribution of wave energy as waves move from the open ocean across a continental shelf to the shoreline. At the location of the 5-meter isobath (bathymetric contour), CGEM uses the information on nearshore wave conditions, computed by SWAN, to calculate angle of incidence, longshore component of wave energy flux, and longshore sediment transport potential (assuming a transport-limited scenario) along the entire reach of the portion of coast being analyzed. Detailed explanations of each function and the individual modules of CGEM are discussed in section 2.

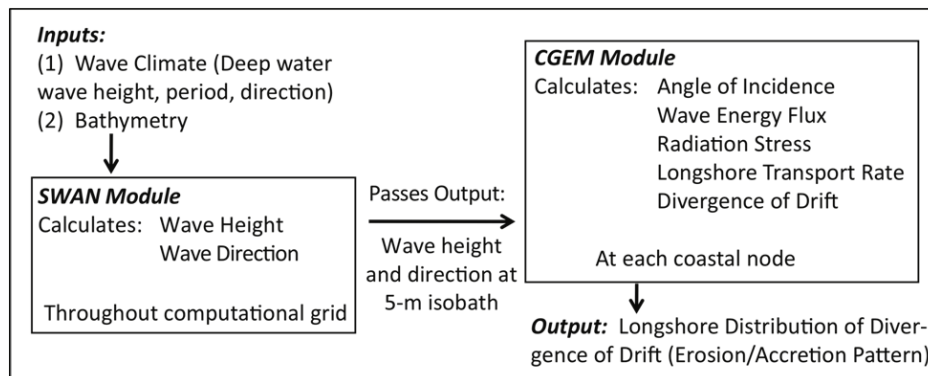


Figure 1-1: Coastal Evolution Model architecture showing the relationship of the SWAN and CGEM modules



## 1.2 Model Inputs

The coastal evolution model, which consists of the wave transformation portion (SWAN) and the longshore sediment transport portion (CGEM), requires only two general inputs: bathymetry and offshore (deep-water) wave conditions. In subsections 1.2.1 and 1.2.2, we describe the details, requirements, and formats of these two general inputs.

### 1.2.1 Bathymetric Data

Ocean bathymetry exhibits a strong control on the direction and rate of wave energy translation. Linear Airy wave theory, used when wave height is much smaller than wavelength and water depth, predicts that wave orbital motions interact to a depth of approximately half the wavelength (Komar, 1998). When the water depth is shallower than half the wavelength, interaction of wave orbitals with the sea floor causes shoaling transformation and refraction of waves. Therefore, the spatial pattern of nearshore wave energy depends strongly on the distribution of seafloor elevation. Bathymetric data are required for SWAN model simulations of wave transformation. Although, SWAN can accept bathymetric data in many formats, these functions are designed to accomodate a grid of bathymetry with a spatial resolution of 3 arc-seconds ( 93 meters latitudinal spacing, 77 meters longitudinal spacing). Bathymetric data can be obtained from the National Geophysical Data Center (NGDC) 3 arc-second U.S. coastal relief model grid database. This database provides coverage of nearshore, shelf, and proximal deep ocean bathymetry for the coterminous U.S. coastline, including Hawaii and Puerto Rico. By using a grid-based bathymetry, changing sea level is a trivial matter performed simply by adding a scalar value to each element of water depths in the bathymetric matrix. This is further simplified by the seamless coverage of the database from offshore to onshore terrain.

### 1.2.2 Offshore Wave Climate

To proscribe wave climate for the SWAN module, information on wave height, period, and direction for the wave field must be provided. For wave height, some representation of the central tendency must be provided; these functions are designed for significant wave height, which is defined as the average of the height of the largest one-third of the waves as measured over a specific time interval. For wave period, a spectrally dominant, peak period and assume a Joint North Sea Wave Project (JONSWAP) distribution of frequency for the remainder of the spectrum is provided (Hasselmann et al., 1973). For the wave direction, a spectrally dominant wave direction and a

“spread angle” where the cosine is calculated to account for the remainder of energy distribution within the directional spectrum is provided. To provide an instantaneous snapshot of the wave field in the region, a stationary SWAN run is conducted using the three specific wave climate variables, as described above. The output result is the complete stationary wave field (values of height and direction) at each grid point within the model domain. According to Airy wave theory, the wave period is not expected to change, but values may differ from the mean, dominant period, as the JONSWAP distribution is used as an input. The summary outputs obtained, calculated from wave energy in the spectral bins, are (1) significant wave height ( $H_s$ ) in deep water, (2) peak (spectrally dominant) wave period of the significant wave height ( $T_s$ ), and (3) peak (spectrally dominant) wave direction ( $\alpha$ ), for a reference deep-water location; a hindcast node in the model domain.

### 1.3 Wave Transformation Modeling

SWAN is written in FORTRAN accompanied by a series of MATLAB .m-files to write the required input files for SWAN (.swn files) and execute both the main grid and nested grid calculations of wave transformation. Main grid calculations refer to the coarsely spaced (ie. 30 arc-second) initial pass SWAN computation (in stationary mode), which solves for the complete field of wave conditions over a large scale grid. This results in a coarse matrix of wave heights, periods, and directions. Once the main grid wave conditions have been computed, these values are used as boundary conditions at the oceanic edges of the nested grid locations. These nested grids may be of significantly higher resolution to show detailed wave transformations along a shoreline.

### 1.4 Longshore Sediment Transport Modeling

The final output from the SWAN component of the coastal evolution model is a complete oceanic grid of significant wave heights and directions ( $H_s$  and  $D$ , respectively) at each wet node within the model domain. The task of the CGEM component of the coastal evolution model is to convert the modeled wave field into the alongshore distribution of regions of sedimentary erosion and accretion, i.e., the locations and magnitudes of erosional “hotspots.” Subsections 1.4.1 through 1.4.8 below provide the details of these calculations and illustrations of the numerical steps necessary to go from a nearshore wave field to a calculation of coastal erosion potential. For the sake of continuity and illustrative purposes, an example case calculation of coastal

erosion potential for an instantaneous set of wave conditions provided in appendix B.

#### 1.4.1 Choosing the 5-meter Isobath

The first step is to select a consistent location at which to query the SWAN modeled wave field. Ideally, this selection should be based on the point of wave breaking, when the shoaling wave releases its energy by breaking and converts the broken wave energy to nearshore currents. However, the depth at which wave breaking occurs is dynamic, being highly dependent on the wave steepness, which has both wave height and period dependence (Kaminsky and Kraus, 1993). Typical ranges for breaking depths are 8 to 2 meter water depth, as calculated by the relationship provided in Komar and Gaughan (1972). To avoid this complication, the 5-meter isobath is selected as the location set to query the SWAN results and discuss, model limitation are discussed in Section 1.5 below. In the CGEM model, this is done by numerical routines, which apply MATLABs contouring algorithm to each of the bathymetric nests provided in order to locate the map view coordinates of the 5-meter isobath. The uncorrected 5-meter isobath locations must then be quality-checked by the user, to remove closed contours in the nearshore, and distant anchor points that MATLAB establishes to reference the reported contour position locations. Some degree of correction must be conducted for each specific nest.

#### 1.4.2 Decimating Along the 5-meter Isobath

After the corrected, 5-meter isobath is chosen for each nest, a consistent spacing along the isobath must be calculated, so as to have an even alongshore distribution of modeled nearshore wave conditions.

#### 1.4.3 Computing Coastal Trends

An accurate measurement of the orientation of the coastline is necessary for the computation of angle of incidence ( $\alpha$ ), or the difference between nearshore wave direction and coast normal direction. This is a fundamental variable in the calculation of both the radiation stress ( $S_{xy}$ ) and the longshore component of wave energy flux ( $P_\ell$ ), also known as the stress-flux factor. Detailed computation of coastal trends for the decimated 5-meter isobaths in each of the nested grids was performed by the following procedure:

- Starting at the northwest-most portion of the isobath, create a subset consisting of a user specified number of adjacent points from the model

*last modified March 9, 2012*

output.

- Fit a linear regression orientation line to the subset, using the widest range spatial orientation (latitude or longitude) as the independent variable. This ensures that shorelines with a more north-south trend are fit with longitude as the independent variable and that shorelines with a more east-west trend are fit with latitudes as the independent variable.
- Record the direction of the normal to the fitted trendline (90 clockwise from trend) as the shore-normal value for the midpoint location of the subset.
- Shift the subset of points downcoast (toward the southeast) by one model output position and repeat the steps above.

#### 1.4.4 Retrieving and Interpolating SWAN Output

The task of obtaining model output at the decimated model output positions is performed by simply conducting a two-dimensional linear interpolation of the SWAN nested grid results for wave height and direction. After retrieval, the alongshore wave heights and directions are smoothed with a 1-km moving average window.

#### 1.4.5 Calculating Angle of Incidence

A major control on longshore sediment rate is the longshore component of wave energy flux. The angle of incidence is of primary importance in the calculation of longshore sediment transport, as will be further discussed in subsection 1.4.7. If wave rays approach the beach at an angle perfectly orthogonal to the trend of the coast, the longshore component of wave energy flux is zero, and there is no net longshore current to drive longshore sediment transport. If wave rays approach the beach at an oblique angle (somewhere between orthogonal and parallel), there is a component of wave energy flux parallel to the shoreline, which drives longshore sediment transport.

The calculation of angle of incidence in the CGEM model is quite straightforward and proceeds as follows: Assuming a north-south trending coastline with land to the east and sea to the west, the angle of incidence is the difference between the nearshore wave direction (azimuth) and the coast normal (azimuth). Two scenarios are described below. In scenario

*last modified March 9, 2012*

A, nearshore waves approach from the northwest ( $D \sim 300^\circ$ ) and the coast normal is approximately due east, referenced by the direction from which the vector originates ( $N \sim 300^\circ$ ), making the angle of incidence ( $\alpha$ ) equal to approximately  $+30^\circ$ . Longshore currents generated for scenario A would be directed southward. In scenario B, nearshore waves approach from the slightly south of west ( $D \sim 265^\circ$ ) and the coast normal is approximately due east, as in scenario A ( $N \sim 270^\circ$ ), making the angle of incidence ( $\alpha$ ) equal to approximately  $-5^\circ$ . Longshore currents generated for scenario B would be directed northward, with decidedly less magnitude.

#### 1.4.6 Calculating Wave Energy Flux

Wave energy flux ( $P$ ), which has dimensions of  $[\frac{\text{mass} \cdot \text{length}}{\text{time}^3}]$  and units of  $[\frac{\text{Watts}}{\text{meter of shoreline}}]$ , and is calculated through the relationship

$$P = ECn = \frac{1}{8} \rho g H^2 C n \quad (1)$$

where  $E$  is wave energy density, which has dimensions of  $[\frac{\text{mass}}{\text{time}^2}]$  and units of  $[\frac{\text{joules}}{\text{meter}^2}]$ ,  $c$  is nearshore wave celerity, which is depth controlled and has dimensions of  $[\frac{\text{length}}{\text{time}}]$  and units of  $[\frac{\text{meters}}{\text{second}}]$ ,  $n$  is ratio of group to individual wave speed ( $\approx 1$  in shallow water) and is dimensionless,  $\rho$  is the density of seawater  $[1024 \frac{\text{kg}}{\text{m}^3}]$ ,  $g$  is gravitational acceleration  $[9.81 \frac{\text{m}}{\text{s}^2}]$ , and  $H$  is nearshore wave height, which has dimensions of  $[\text{length}]$  and units of  $[\text{meters}]$ , as computed by the SWAN portion of the model and interpolated along the decimated 5-meter isobath.

The longshore component of wave energy flux ( $P_\ell$ ) is calculated by simply multiplying the wave energy flux ( $P$ ) by the trigonometric functions that provide the component parallel to shore

$$P_\ell = P \sin \alpha \cos \alpha \quad (2)$$

where the sine and cosine terms result from the tensor transformation of the onshore flux of longshore directed momentum that is embedded in  $P_\ell$  (Longuet-Higgins and Stewart, 1964). Procedure for calculation of angle of incidence ( $\alpha$ ) is described in Section 1.4.5, above.

#### 1.4.7 Calculating Divergence of Drift

The formulation of longshore sediment transport calculation used in the CGEM model comes from the sediment-transport theories of Bagnold (1963),

*last modified March 9, 2012*

Inman and Bagnold (1963), and Komar and Inman (1970). These theories utilize the concept of immersed-weight sediment transport rate ( $I_\ell$ ) to account for density of sediment grains,

$$I_\ell = K_\ell P_\ell = K_\ell [P \sin \alpha \cos \alpha] \quad (3)$$

where  $K_\ell$  represents a dimensionless coefficient of proportionality, as defined by Komar and Inman (1970) (Komar and Inman, 1970), and is set to a default value of 0.8 for all of the numerical experiments described below. Immersed-weight transport rate is converted to volumetric transport rate through the relationship

$$Q_\ell = \frac{I_\ell}{(\rho_s - \rho_w)gN_o} \quad (4)$$

where  $\rho_s$  and  $\rho_w$  are densities of quartz sediment ( $2650 \frac{kg}{m^3}$ ) and seawater ( $1024 \frac{kg}{m^3}$ ), respectively,  $g$  is the gravitational acceleration constant ( $9.81 \frac{m}{s^2}$ ), and  $N_o$  is the volume concentration of solid grains, set to 0.6, for all numerical experiments described below.

The calculation of the volumetric rate of longshore sediment transport yields a rather noisy result. To obtain a more reasonable estimate of local trends in longshore sediment transport, we smooth the calculations of longshore transport with a 1-km moving average window.

The last remaining step to obtain volumetric estimates of gradients in longshore transport (also known as the divergence of drift) is to perform a discretized differential of volumetric rate of longshore sediment transport with respect to alongshore position.

$$\nabla \cdot Q_\ell = \frac{\delta Q_\ell}{\delta \ell} \quad (5)$$

The results of this calculation are out of phase with longshore sediment transport, as expected, illustrating regions where longshore sediment transport rate reaches a local maxima or minima correspond to regions where the divergence changing from negative to positive, or positive to negative, respectively.

## 1.5 Model Limitations

We acknowledge the assumption that the 5-meter isobath is representative of nearshore wave conditions should be taken into account appropriately.

*last modified March 9, 2012*

Under conditions of milder wave fields, wave breaking will be landward of the 5-meter isobath, and for similar considerations, wave breaking under conditions of highly energetic wave fields will be seaward of the 5-meter isobath location. However, for most wave fields the break point will be landward of the 5-meter isobath, and in these cases, we can rely on the conservation of wave energy flux to protect our assumption.

It should be noted that the calculations of divergence of drift, described in Section 1.4.7, are potential divergence of drift. Assuming a transport-limited longshore drift scenario, we are calculating the total amount of longshore transport of sediment if an unlimited supply was available. This may not be a likely case for some areas, such as the Southern California coast, as river-damming and the arid environment tends to make the littoral system supply-limited in many places. Where the wave energy demands on sediment transport exceed the supply, bedrock platforms will be exposed and erosion of sea cliffs will proceed with greater efficiency.

## 1.6 References

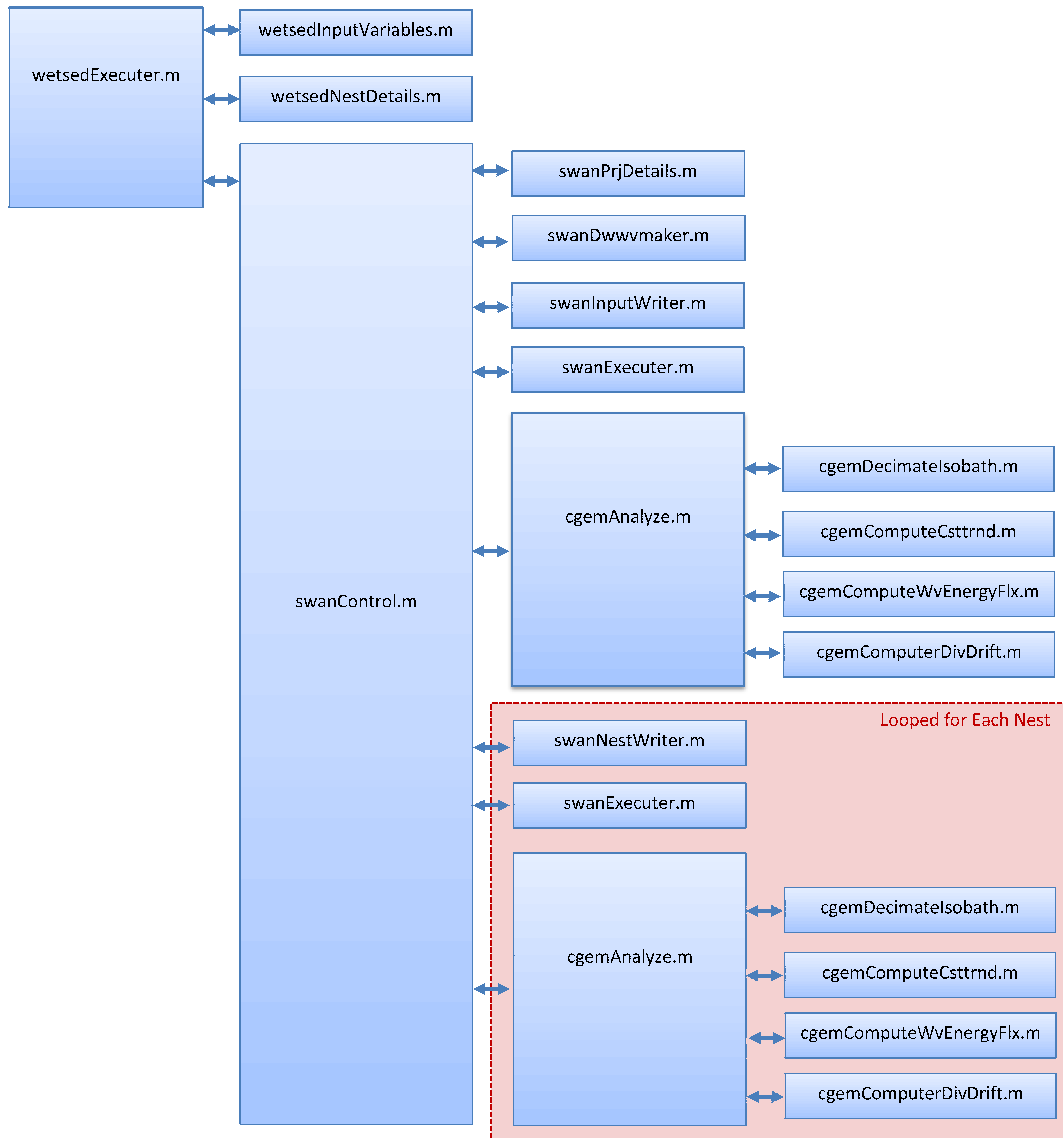
### References

- Bagnold, R. (1963). *Beach and Nearshore Processes - Part 1, Mechanics of marine sedimentation*, volume 3, pages 507–528. New York and London, John Wiley and Sons.
- Booij, N., Ris, R. C., and Holthuijsen, L. H. (1999). A third-generation wave model for coastal regions - 1. model description and validation. *Journal of Geophysical Research-Oceans*, 104(C4):7649–7666.
- Hasselmann, K., Barnett, T. P., Bouws, E., Carlson, H., Cartwright, D. E., Enke, K., Ewing, J. A., Gienapp, H., Hasselmann, D. E., Kruseman, P., Meerburg, A., Mller, P., Olbers, D. J., Richter, K., Sell, W., and Walden, H. (1973). Measurements of wind-wave growth and swell decay during the joint north sea wave project (jonswap). *Ergänzungsheft zur Deutschen Hydrographischen Zeitschrift Reihe*, 8(12):95.
- Inman, D. L. and Bagnold, R. A. (1963). Littoral processes. In Hill, M. N., editor, *The Sea*, volume 6, pages 529–553. Wiley-Interscience, New York.
- Kaminsky, G. and Kraus, N. C. (1993). Evaluation of depth-limited wave breaking criteria. In *Proceedings of 2nd International Symposium on Ocean Wave Measurement and Analysis, Waves 93: New York ASCE*, pages 180–193.
- Komar, P. D. (1998). *Beach Processes and Sedimentation*. Prentice-Hall, Inc., second edition.
- Komar, P. D. and Gaughan, M. K. (1972). Airy wave theory and breaker height prediction. In *Proceedings of the 13th Coastal Engineering Conference*, pages 405–418. Amer. Soc. Civil Engrs.
- Komar, P. D. and Inman, D. L. (1970). Longshore sand transport on beaches. *Journal of Geophysical Research*, 75(30):5914–5927.
- Longuet-Higgins, M. S. and Stewart, R. W. (1964). Radiation stresses in water waves; a physical discussion, with applications. *Deep-Sea Research*, 11:529–562.



## 2 Function Descriptions

### 2.1 Model Flow



## 2.2 wetsedExecuter.m

This Matlab script calls all the other programs and loops through each set of bathymetries.

**Called by:** *none*

**Calls Functions:** wetsedInputVariables.m, wetsedNestDetails.m, swanControl.m

---

### 2.2.1 Inputs and Outputs

#### Input Variables

*none*

#### Output Variables

*none*

---

### 2.2.2 Steps

1. Runs wetsedInputVariables.m to get user input wave condition and directory information
2. Runs wetsedNestWriter.m to get user grid information
3. Begin to loop through SwanControl.m

## 2.3 swanControl.m

Main controlling Function for each individual SWAN run.

**Called by:** wetsedExecuter.m

**Calls Functions:** swanPrjDetails.m, swanDwwvmker.m,  
swanInputWriter.m, swanNestWriter.m, swanExecuter.m, cgemAnalyze.m

---

### 2.3.1 Inputs and Outputs

#### Input Variables

· prj_s	· base_dir	· input_dir	· projectName
· num_processes	· Hs_range	· T_range	· Dir_range
· spread	· nest2run	· bathy_grd_name	· wetsed_ver
· mx	· my	· nest_names	· xp
· yp	· xlen	· ylen	· alp
· user			

#### Output Variables

*none*

---

### 2.3.2 Steps

1. Retrieves Computations Grid data Based on prj\_s from swanPrjData.m
2. Begins MPI usage
3. Constructs a synthetic time series of Deep Water Wave Conditions by calling swanDwwvMaker.m
4. Creates Output file Directories for each set of H,T,Dir combination in /SwanOutput/
5. Main Grid: Calls swanInputWriter.m to create the text input file for SWAN, executes SWAN with swanExecuter.m for main grid, then executes cgemAnalyze.m for main grid
6. Nest Loop: Calls swanNestWriter.m to create the text input file for SWAN nested grids, executes SWAN with swanExecuter.m for each nested grid, then executes cgemAnalyze.m for each nest
7. Stops MPI usage

*last modified March 9, 2012*

## 2.4 swanPrjDetails.m

This function retrieves Grid dimensions.

**Called by:** SwanControl.m

**Calls Functions:** *none*

---

### 2.4.1 Inputs and Outputs

#### Input Variables

· prj\_s                      · input\_dir

#### Output Variables

· dxinp                      · dyinp                      · num\_cols  
· num\_rows

---

### 2.4.2 Steps

1. Loads the file ‘prj\_s\_grid\_vars.mat’
2. Loads variables: dxinp, dyinp, num\_cols, num\_rows, and gridspacing
3. Returns the dimensions based on this mat file content

## 2.5 swanDwwvmaker.m

This constructs a synthetic time series of deep water wave conditions.

**Called by:** SwanControl.m

**Calls Functions:** *none*

---

### 2.5.1 Inputs and Outputs

#### Input Variables

· batch_num	· input_dir	· first_runnum
· Hs	· T	· Dir

#### Output Variables

· run\_nums · Hs\_all · T\_all  
· Dir\_all

---

### 2.5.2 Steps

1. Loops through all combinations of H, T, and Dir to make a list of run scenarios

## 2.6 swanInputWriter.m

This function writes the text file used by SWAN to define variables and filenames used for the run.

**Called by:** SwanControl.m

**Calls Functions:** swanCalcIJVals.m

---

### 2.6.1 Inputs and Outputs

#### Input Variables

· run_num_s	· Hs	· T	· Dir
· nest_id	· num_rows	· num_cols	· dyinp
· dxinp	· bathy_grd_name	prj_s	· new_dir
· mx	· my	· nest_names	· xp
· yp	· xlen	· ylen	· alp
· projectName	· spread	· wetsed_ver	· user
· cgridscale			

#### Output Variables

*none*

---

### 2.6.2 Steps

1. Calculate Grid Height and Width
2. Define values to be used with computational grid (CGRID REGular and CIRcle statements)
3. Define values to be used with bathymetric grid (INPgrid REGular statements)
4. Define values to be used with boundary conditions (BOUNDspec SEG UNIFORM PAR commands) by calling swanCalcIJVals.m.
5. Define output grid information
6. Creates SWAN input file named: swan\_ 'prj\_s' \_ 'run\_num\_s' \_ 'nest\_id'.swn
7. Write header information and simulation conditions
8. Write lines in main .swn input file to create nested grids (with a loop)

*last modified March 9, 2012*

## 2.7 swanCalcIJVals.m

This function defines the boundary condition segments used by SWAN

**Called by:** swanInputWriter.m

**Calls Functions:** *none*

---

### 2.7.1 Inputs and Outputs

#### Input Variables

· *prj\_s*                      · *mx\_c*                      · *my\_c*

#### Output Variables

· *i\_vals*                      · *j\_vals*

---

### 2.7.2 Steps

1. Defines the x and y points that are defined as a percentage of the total grid
2. Defines these as they are specified for each *prj\_s*

## 2.8 swanNestWriter.m

This function prepares the nested input files for SWAN.

**Called by:** SwanControl.m

**Calls Functions:** *none*

---

### 2.8.1 Inputs and Outputs

#### Input Variables

· run_num_s	· Hs	· T	· Dir
· nest_id	· nest_str	· xlenn	· bathy_grd_params
· prj_s	· new_dir	· mxn	· myn
· xpn	· ypn	· ylenn	· bathy_grd_name
· alpn	· projectName	· spread	

#### Output Variables

*none*

---

### 2.8.2 Steps

1. Write .swn file for SWAN nest grids with variables input into the function



## 2.9 swanExecuter

This function initiates the SWAN program to run with the text file created in swanInputWriter.m

**Called by:** swanControl.m

**Calls Functions:** *none*

---

### 2.9.1 Inputs and Outputs

#### Input Variables

· prj\_s                      · run\_num\_s                      · nest\_id  
· num\_processes

#### Output Variables

*none*

---

### 2.9.2 Steps

1. Runs SWAN through the operating system and defined Multiprocessor data
2. SWAN creates a file out\_grid.mat with SWAN results

## 2.10 cgemAnalyze.m

This is the main function to control the calculation CGEM component of the model.

**Called by:** wetsedExecuter.m

**Calls Functions:** cgemDecimateIsobath.m, cgemComputeCsttrnd.m,  
swanNanFix.m, cgemSmoother.m, cgemComputeAngleInc.m,  
cgemComputeWvEnergyFlx.m, cgemComputeDivDrift.m

---

### 2.10.1 Inputs and Outputs

#### Input Variables

· prj\_s                      · H                      · T  
· Dir                      · wetsed\_ver              · user  
· nest\_str

#### Output Variables

*none*

---

### 2.10.2 Steps

1. Enter directory information, wave coeff, and grid parameters, including:
  - input\_dir
  - output\_dir
  - isobath\_dir
  - sm\_win\_sz
  - coast\_sm\_win\_sz1
  - coast\_sm\_win\_sz2
  - dec\_spc
  - K
2. Load isobath and decimate with cgemDecimateIsobath.m
3. Measure coastal trends along decimated 5m isobath with cgemComputeCsttrnd.m

*last modified March 9, 2012*

4. Retrieve swan results for for the current bathymetry
5. Interpolate swan results along decimated 5m isobath, remove NaNs, and smooth with `cgemSmoother.m`
6. Calculate angle of incidence along 5m isobath for regular and smoothed heights
7. Compute `WvEnrgy`, `Wv.EnrgyFlx`, `Sxy`, and `StrssFlxFactor` with `cgemComputeWvEnergyFlx.m`
8. Compute divergence of drift along 5m isobath with `cgemComputeDiv-Drift.m`
9. Save Output to a mat file

## 2.11 cgemDecimateIsobath.m

This function converts irregularly spaced isobath coordinates into a uniformly spaced array.

**Called by:** cgemAnalyze.m

**Calls Functions:** *none*

---

### 2.11.1 Inputs and Outputs

#### Input Variables

· Dist\_05                      · x\_05                      · y\_05  
· dec\_spc

#### Output Variables

· Dist\_05\_i                      · x\_05\_i                      · y\_05\_i

---

### 2.11.2 Steps

1. Locates area with inconsistent spacing in the 5m isobath
2. Uses a 1-D interpolation to equally space the coordinates

## 2.12 cgemComputeCsttrnd.m

This function determines the trend (orientation) of the 5m depth contour

**Called by:** cgemAnalyze.m

**Calls Functions:** *none*

---

### 2.12.1 Inputs and Outputs

#### Input Variables

· x\_05\_i                      · y\_05\_i                      · winsize

#### Output Variables

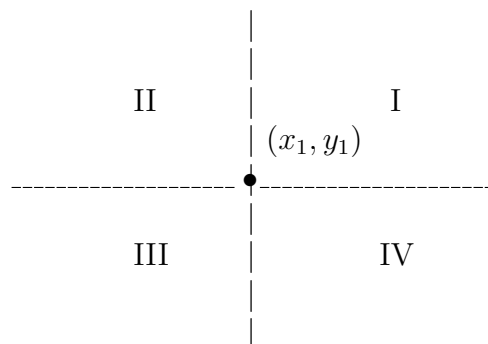
· CT\_c                      · CT\_u                      · Q  
 · CN                      · m\_h                      · m\_v  
 · range\_x                · range\_y

---

### 2.12.2 Steps

1. Calculate the alongshore “slope”
2. Determine which direction the last fit point is with respect to the first fit point  $(x_1, y_1)$ , and assign a quadrant accordingly
3. Calculate alongshore uncorrected coastal trend
4. Calculate corrected coastal trend and coast normal.

**Coastal Trend Quadrants:**



*last modified March 9, 2012*

## 2.13 cgemComputeWvEnergyFlx.m

This function computes longshore component of wave energy flux along the 5m isobath.

**Called by:** cgemAnalyze.m

**Calls Functions:** *none*

---

### 2.13.1 Inputs and Outputs

#### Input Variables

· x\_05\_i                      · y\_05\_i                      · H\_05\_i  
· AL\_05\_i

#### Output Variables

· E\_05\_i                      · P\_05\_i                      · RS\_05\_i  
· SF\_05\_i

---

### 2.13.2 Steps

1. Compute wave energy

$$E = \frac{1}{8} \rho g H^2$$

2. Compute wave celerity

$$c = \sqrt{gh}$$

3. Compute energy flux

$$P = ECn$$

4. Compute radiation stress

$$S_{xy} = En \cos \theta \sin \theta$$

5. Compute the longshore component of wave energy flux

$$P_\ell = P \cos \theta \sin \theta$$

*last modified March 9, 2012*

## 2.14 cgemComputeDivDrift.m

This function computes the divergence of drift along the 5m isobath.

**Called by:** cgemAnalyze.m

**Calls Functions:** *none*

---

### 2.14.1 Inputs and Outputs

#### Input Variables

· SF\_05\_i                      · dec\_spc                      · K

#### Output Variables

· IR\_05\_i                      · QR\_05\_i                      · DD\_05\_i

---

### 2.14.2 Steps

1. Compute transport rate

$$I_\ell = KP_\ell$$

2. Compute volumetric transport rate

$$Q = \frac{KP_\ell}{(\rho_s - \rho)gN_o}$$

3. Compute divergence of drift

$$\nabla \cdot Q_\ell = \frac{\delta Q_\ell}{\delta \ell}$$

### 3 User Input Files to Modify

The user should modify the contents of the following functions for a new simulation:

- wetsedInputVariables.m
- wetsedNestDetails.m

The user should modify the contents of the following mat and text files for a new simulation:

- *prj\_s\_grid\_vars*.mat
- *prj\_s\_swan*.txt
- *Isobath\_prj\_s*.mat



## 4 User Defined Input Files

### 4.1 Main Grid Variables MAT File

This input file contains details of the nest grids for each nest

Filename: *prj-s\_grid.vars.mat*

Loaded in:  
swanPrjDetails.m

contains variables:  
dxinp            dyinp            gridspaceing  
num\_cols        num\_rows

### 4.2 Bathymetry Input Text Files

This input file contains the bathymetry height at each grid location, used directly by SWAN

Filename: *prj-s\_swan.txt*

Loaded in:  
SWAN program

contains variables:  
Z

### 4.3 5m Isobath Location MAT File

This input file contains the locations of the 5m isobath for the given *prj-s*

Filename: *Isobath\_prj-s.mat*

Loaded in:  
cgemAnalyze.m

contains variables:  
x\_05            y\_05

## 5 Definition of Variables

<b>AI_05_i</b>	This is the angle, $\theta$ , at which wave rays approach the shoreline
<b>alp</b>	This is a varibale associated with the nested grids (??)
<b>base_dir</b>	The directory that the SWAN output will be directed to.
<b>bathy_grd_name</b>	The name of the text file that contains the depth values for all (x,y) points of the computations grid
<b>bathy_grd_params_s</b>	A string of grid bathymetry input variables including xpinp, ypinp, alpinp, mxinp, myinp, dxinp, dyinp used to create the input text file created for SWAN
<b>CN</b>	Corrected coastal trend +90 degrees (??)
<b>CT_c</b>	The coastal trend direction in degrees corrected by assigning a quadrant
<b>CT_u</b>	The uncorrected coastal trend in degrees
<b>DD_05_i</b>	The divergence of drift along the 5m isobath
<b>dec_spc</b>	The grid spacing defined by the user in cgemAnalyze-Geometric.m
<b>Dfix</b>	Wave direction found after removing NaNs with the SwanNanFix.m function
<b>Dir</b>	Wave Direction, North = $0^0$ increasing clockwise
<b>Dir_range</b>	An array containing a variety of wave directions input values specified by the user
<b>Dist_05</b>	An array containing the distances between grid points along the 5m isobath before the are spaced at even increments

*last modified March 9, 2012*

<b>Dist_i</b>	An array containing the distances between grid points along the 5m isobath after they have been interpolated to become spaced at even increments
<b>dxinp</b>	The distance between grid spacing in the x direction of the input bathymetry text file
<b>dyinp</b>	The distance between grid spacing in the y direction of the input bathymetry text file
<b>E_05_i</b>	Energy Flux along the interpolated 5m isobath
<b>gridspacing</b>	A string identifier that specifies that the grid spacing is in either cartesian or arc second mode dependent on <i>prj-s</i> .
<b>H</b>	Wave height before the NaN have been removed from the SWAN output
<b>H_05_i</b>	The wave heights along the interpolated 5m isobath
<b>Hfix</b>	The SWAN wave heights with NaNs removed
<b>Hs</b>	The significant wave height given to SWAN as initial conditions along the specified boundaries
<b>Hs_range</b>	An array containing a range of wave height input values specified by the user
<b>i_vals</b>	The point numbers of the x coordinates of the boundary condition segments for SWAN
<b>input_dir</b>	The user defined directory in which the input files for the SWAN run are located
<b>IR_05_i</b>	The transport rate ( $I_\ell$ ) calculated along the interpolated 5m isobath

<b>j_vals</b>	The point numbers of the y coordinates of the boundary condition segments for SWAN
<b>K</b>	The dimensionless coefficient of proportionality used to calculate immersed weight sediment transport rate. This is a user defined values, but default value is set to 0.8
<b>m_h</b>	The horizontal component of slope used to find the coastal trend direction of the 5m isobath
<b>m_v</b>	The vertical component of slope used to find the coastal trend direction of the 5m isobath
<b>method</b>	A string signifying the type of interpolation used to replace values removed by swanNanFix.m, method can be 'spline' , 'cubic' or 'linear'
<b>mx</b>	An array containing the number of mesh points in the x direction for each nested grid
<b>mxo</b>	The number of meshes in the x direction of the main grid
<b>my</b>	An array containing the number of mesh points in the y direction for each nested grid
<b>myo</b>	The number of meshes in the x direction of the main grid
<b>nest_id</b>	A string containing a nest number used in file naming
<b>nest_names</b>	An array of strings containing the names of the nest grids
<b>nest2run</b>	A user defined array of numbers that indicates what nested grids to run
<b>new_dir</b>	The output directory which is created out of <i>prj-s</i> , <i>Hs</i> , <i>Dir</i> , <i>T</i> variables in which to put the SWAN run results

---

<b>num_cols</b>	The number of x grid points in the bathymetry text input file
<b>num_processes</b>	The user defined number of processors in which to run MPI with
<b>num_rows</b>	The number of y grid points in the bathymetry text input file
<b>P_05_i</b>	The array of wave energy flux values along the interpolated 5m isobath
<b>prj_s</b>	The 3 letter string defined by the user as the project name
<b>projectName</b>	A string that represents the name of the project to be printed on the SWAN input text file, does not influence the SWAN output results.
<b>Q</b>	The integer identifying the quadrant in which the coastal trend of the 5m isobath is directed toward.
<b>QR_05_i</b>	The volumetric transport rate along the interpolated 5m isobath
<b>range_x</b>	The range of x values in the x_05_i term
<b>range_y</b>	The range of y values in the y_05_i term
<b>RS_05_i</b>	The radiation stress along the interpolated 5m isobath
<b>run_num_s</b>	The number associated with the current swan run, used as an identifier in the file name
<b>SF_05_i</b>	The stress flux factor along the interpolated 5m isobath
<b>spread</b>	The coefficient of directional spreading; $\cos^m(\Theta)$ distribution is assumed.
<b>T</b>	The wave period given as an input condition along the boundary segments for SWAN

*last modified March 9, 2012*

---

<b>T_range</b>	An array containing a range of wave periods input values specified by the user
<b>user</b>	The name of the user used for directory creation
<b>winsize</b>	An integer representing the number of points used to estimate the coastal trend
<b>x_05</b>	An array containing the x grid points along the 5m isobath before they are spaced at even increments
<b>x_05_i</b>	An array containing the x grid points along the 5m isobath once they have been spaced at even increments
<b>xlen</b>	The x direction length the grid in meters, specified in an input mat file containing grid information
<b>xp</b>	The lower left origin x point for a nested grid relative to the main grid, specified in an input mat file containing grid information
<b>y_05</b>	An array containing the y grid points along the 5m isobath before they are spaced at even increments
<b>y_05_i</b>	An array containing the y grid points along the 5m isobath once they have been spaced at even increments
<b>ylen</b>	The y direction length the grid in meters, specified in an input mat file containing grid information
<b>yp</b>	The lower left origin y point for a nested grid relative to the main grid, specified in an input mat file containing grid information

## A An Example using Simplified Geometric Coastal Shelves

### A.1 Introduction

This example is an illustration of how to use `WetSed` on a simple system of synthetic coastal shelves created by the user.

### A.2 Bathymetric Data

There are six synthetic coastal shelves that have been created for this section. All six consist of a sloping section from 20m above sealevel to 120m depth, over a crossshore span of 20km. At the 20km offshore point an additional increased planar slope occurs from 120m depth to 1500m depth over a distance of 20km more. The total alongshore distances in all of these bathymetric grids are 50km. The details of the bathymetries are given below and the first shown in figure A-1. The sloping portion of between 20km and 40km offshore is uniform throughout all the grids and will not be specifically addressed in the sections below. Note that in all of the examples below depth is taken to be a positive value, and all units of length are in meters unless otherwise specified.

#### A.2.1 Planar Sloping Landscape

This is the most simple case and it is the one on which all the other grids are built, shown in Figure A-1(a). It uses the simple equation listed below.

$$Z_{planar} = 0.007Y - 20$$

#### A.2.2 Sinusoidal Coastline Decaying with Depth

The second grid is a sinusoidal grid which decays in amplitude with water depth. The shoreline features a sinusoidal shaped cusped trend, shown in Figure A-1(b). The equation below was used to create the grid.

$$Z = 10 \sin\left(\frac{X}{3000}\right)\left(1 - \frac{Y}{Y_{max}}\right) + Z_{planar}$$

### A.2.3 Exponentially Decaying Depth

The third grid has an exponentially decaying depth, which creates a parallel depth contours, but with more severe slope at the coastaline than at deep ocean depths, shown in Figure A-1(c). Below are the equations used to create this grid.

$$amp = \frac{(20+120)}{e^{-1}}$$

$$Z = \frac{-amp}{2}e^{(-Y/Y_{max})} + 185$$

### A.2.4 Planar Slope with a Headland

The fourth grid has a planar slope with a gaussian shaped headland at the shoreline that creates a promotory into the water, shown in Figure A-1(d). Below is the set of equation used to create this landscape.

$$d = 5\sqrt{(X - \frac{X_{max}}{2})^2 + (Y - \frac{Y_{max}}{2})^2}$$

$$Z = Z_{planar} - 30e^{(-d/50000)^2}$$

### A.2.5 Planar Slope with a Canyon

This grid is constructed in a similar method to the fourth grid, but with a gaussian shaped depression included in the planar landscape, shown in Figure A-1(e). It uses similar equations to create the grid, as shown below.

$$d = 5\sqrt{(X - \frac{X_{max}}{2})^2 + (Y - \frac{Y_{max}}{2})^2}$$

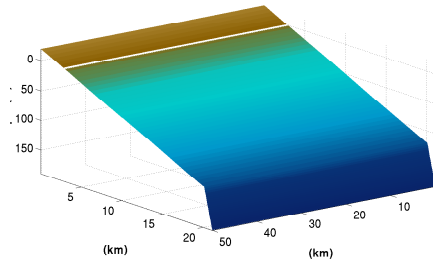
$$Z = Z_{planar} + 20e^{(-d/50000)^2} - 10$$

### A.2.6 Planar Slope with a Hyperbolic Tangent Alongshore Trend

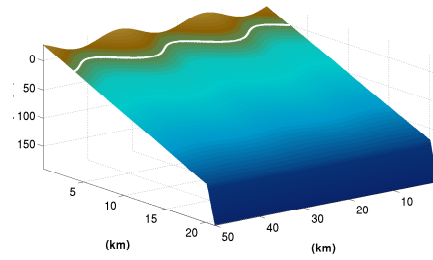
This grid has a hyperbolic tangential trend along the coastline, which creates a protruding section of coast. This grid presents a situation with two different slopes that converge gradually, shown in Figure A-1(f). The equations below were used to create this grid.

$$Z = Z_{planar} + \frac{1}{20} \tanh(X - \frac{X_{max}}{2})(1 - \frac{Y}{Y_{max}})$$

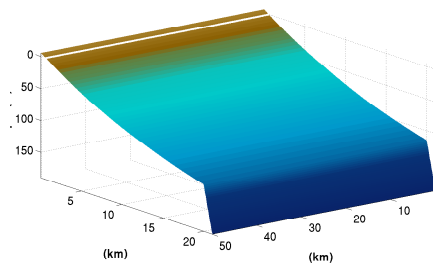




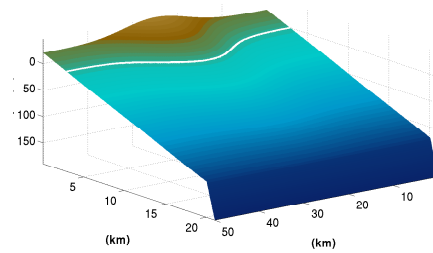
(a) Planar



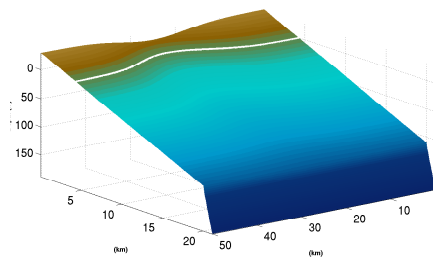
(b) Sinusoidal



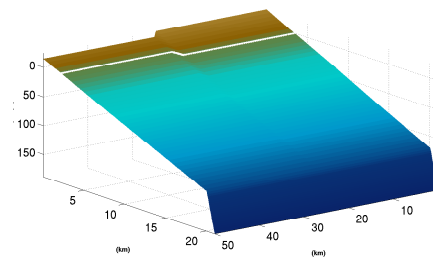
(c) Exponentially Decaying



(d) Planar with a Headland



(e) Planar with a Canyon



(f) Hyperbolic Tangent

Figure A-1: The six simple geometric coastal shelves, each shown up to 200m water depth.

### A.3 Wave Climate

The wave climate used in this example was picked arbitrarily since these grids do not represent a real geographic location. The significant wave height ( $H_s$ ) is 2m, the wave period ( $T$ ) is 10sec, the direction ( $Dir$ ) is  $135^\circ$  with a direction wave spreading (spread) of 15. In the plots of this section  $0^\circ$  is taken to be toward the bottom of the paper, increasing clockwise.

### A.4 Wave Transformation Results

For a simple case such as these simple geometric continental shelves, which involves no nested grids, the input file shown below is created by *WetSed* and is all the instruction that is needed for SWAN to run the simulation. Note it simply needs boundary conditions and a bathymetry files as inputs to the system.

```
!      PROJECT test, Run ID Number: 101849main
!      Run Clock: 07-Jul-2009 16:02:10
!      wave conditions: 2 m, 10 s, 135 deg, (15 deg spread)
!
!
SET NAUTical
MODE STATIONARY TWODimensional
COORDinates CARTesian
CGRID REGular 0 0 0 50000 40100 501 402 CIRCLE 36 0.05 1
INPgrid BOTtom REGular 0 0 0 500 401 100 100 EXC -99999
READinp BOTtom 1 '/SwanInput/bathy_grids/pln_swan.txt' 1 0 FREE
BOUND SHAPespec JONswap 3.3 PEAK DSPR DEGR
BOUNDspec SEG IJ 0 361 0 0 0 501 0 501 0 501 361 CONSTANT PAR 2 10 135 15
OFF QUAD
GROUP 'compgrid' SUBGRID 0 501 0 402
BLOCK 'compgrid' NOHEADER 'out_grid.mat' HSIGN DIR XP YP DEPTH TMM10
COMPUTE
STOP
```

Figure A-2: Sample of a simple SWAN input file

Below in figure [A-3](#) are the outputs of wave heights and directions that have been transformed by SWAN over each of the six grids described in section [A.2](#) with the wave input conditions given in section [A.3](#). The low wave heights observed at the edge represent the area in which the boundary conditions are not applied, this causes a shadow to appear. In order to reduce errors around the outer portions of the grid a nested grid could be used.

Figure [A-4](#) shows a comparison between the waves computed by SWAN over the planar sloping grid described in section [A.2.1](#) and the other grids

of which are created based on the same planar shape. The percentage is calculated, the positive warm portions of the images represent an area of higher wave heights when compared to the planar, and the cooler colors represent lower wave heights.

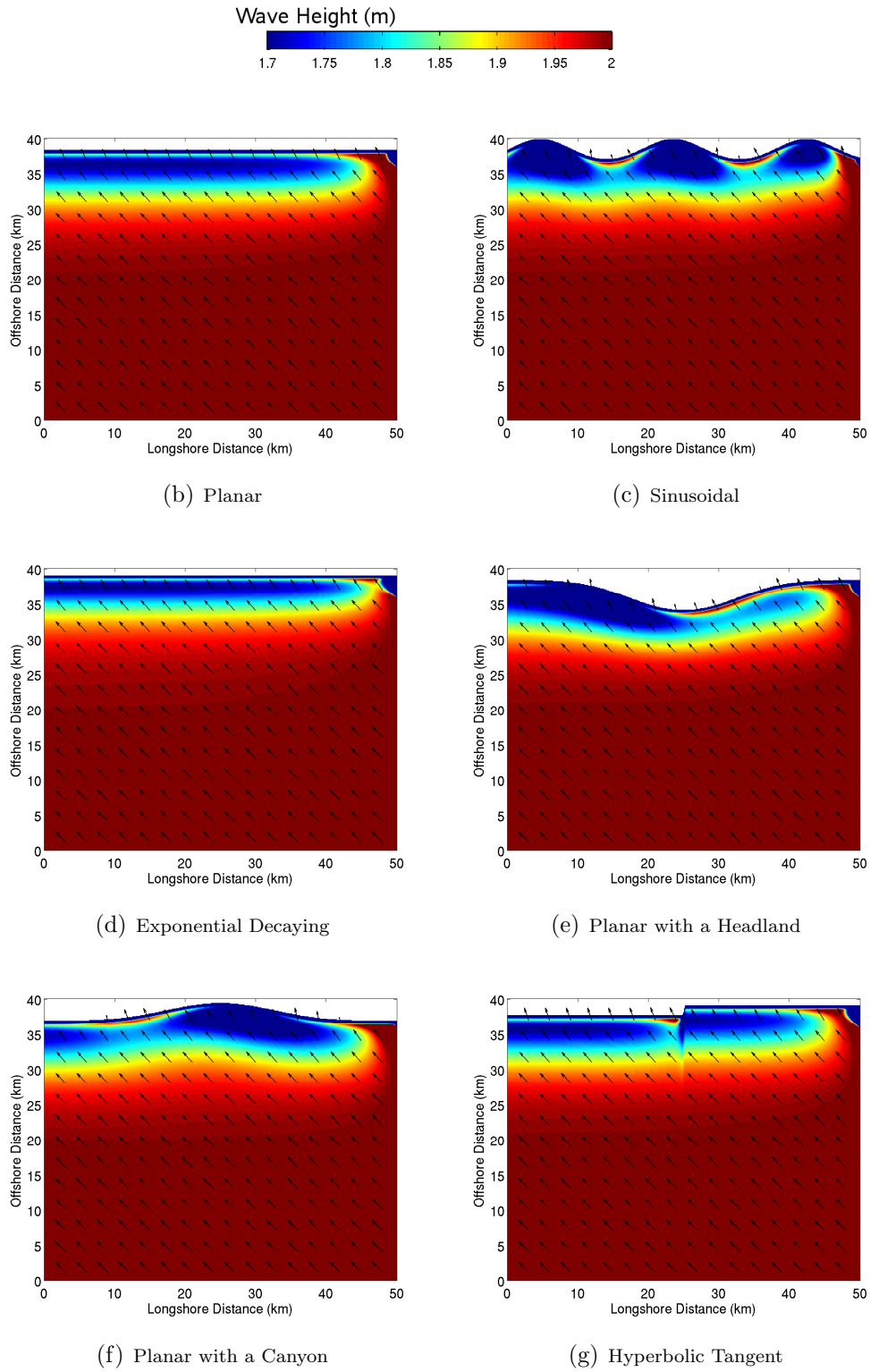


Figure A-3: Wave Heights and Directions computed by SWAN for the 2012 conditions  $H_s=2\text{m}$ ,  $T=10\text{sec}$ , and  $\text{Dir}=135^\circ$  for six different bathymetries.

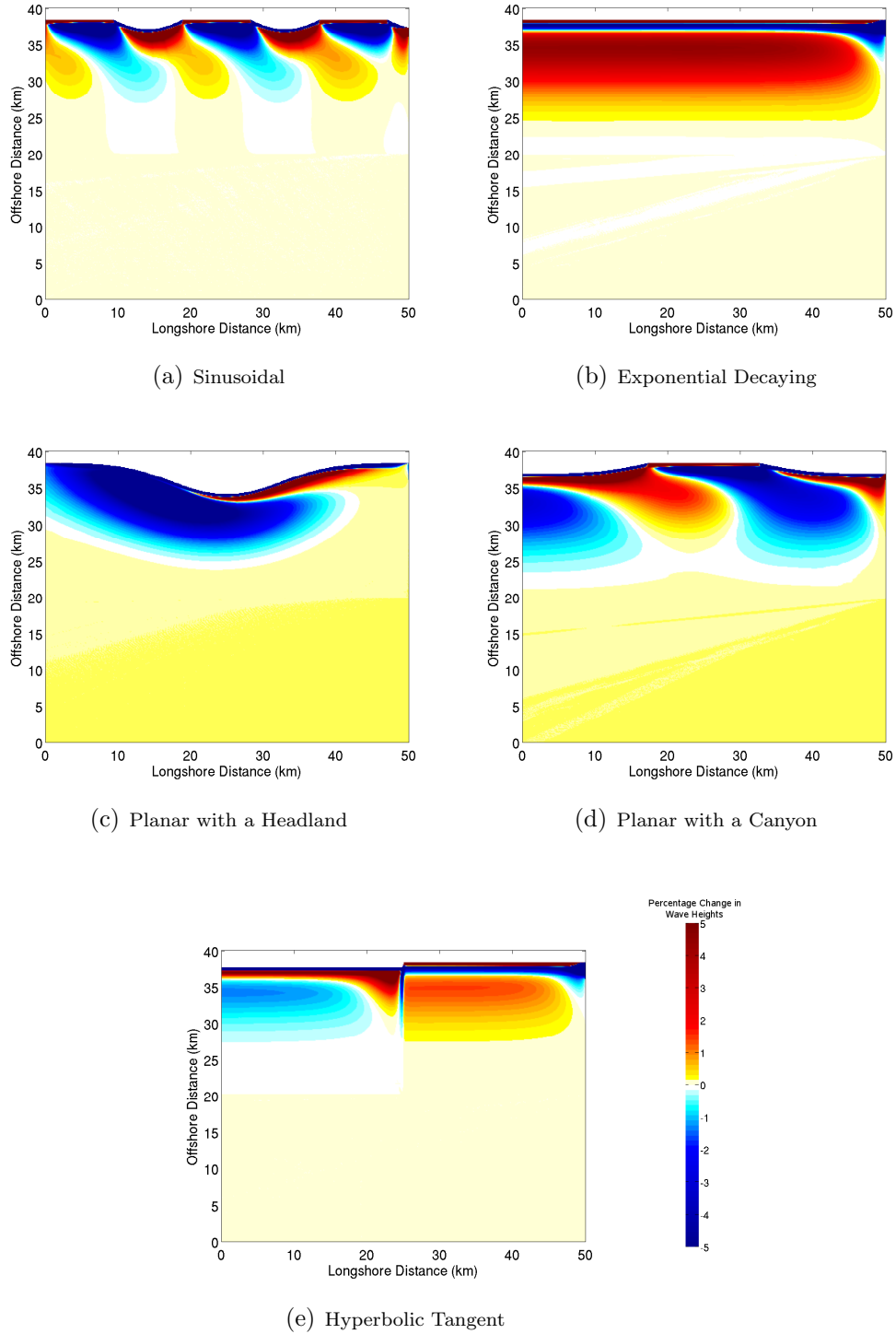


Figure A-4: The percentage change in wave heights from a purely planar continental shelf for each of the five simple geometric bathymetries detailed in section A.2.

*last modified March 9, 2012*

## A.5 Longshore Sediment Transport Model Output

The results that have been acquired from SWAN are now analyzed to show potential sediment transport rates caused by the wave heights and directions. The basic steps with which these are found include 1) locating the 5m isobath 2) interpolate to the 5m isobath to achieve an evenly spaced array 3) find the wave heights and directions on those interpolated points 4) find the angle of incidence of the wave with respect to the shoreline, and 5) calculate the wave energy flux, longshore transport rates, and divergence of drift. Section 1.4 details the steps and equations used in these calculations. The figures A-5 through A-10 show the wave heights, wave direction, sediment transport rates, and divergence of drift along the interpolated 5m isobath for all six simple bathymetric geometries.

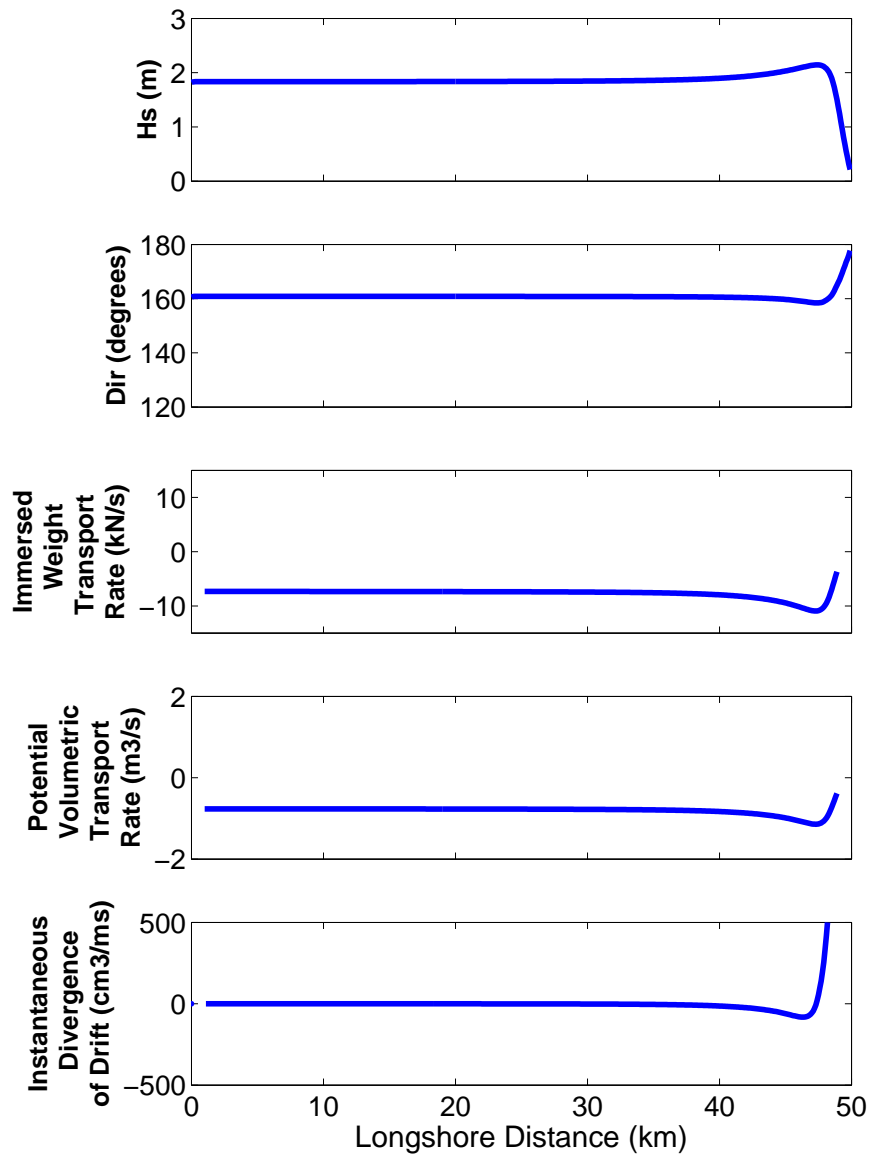


Figure A-5: The step wise calculation of wave height and modeled wave direction along the 5m isobath, and the calculation of longshore sediment transport rates and divergence of drift for the planar sloping continental shelf bathymetry (described in section A.2.1) using the input conditions of  $H_s=2\text{m}$ ,  $T=10\text{sec}$ , and  $Dir=135^\circ$ .

*last modified March 9, 2012*

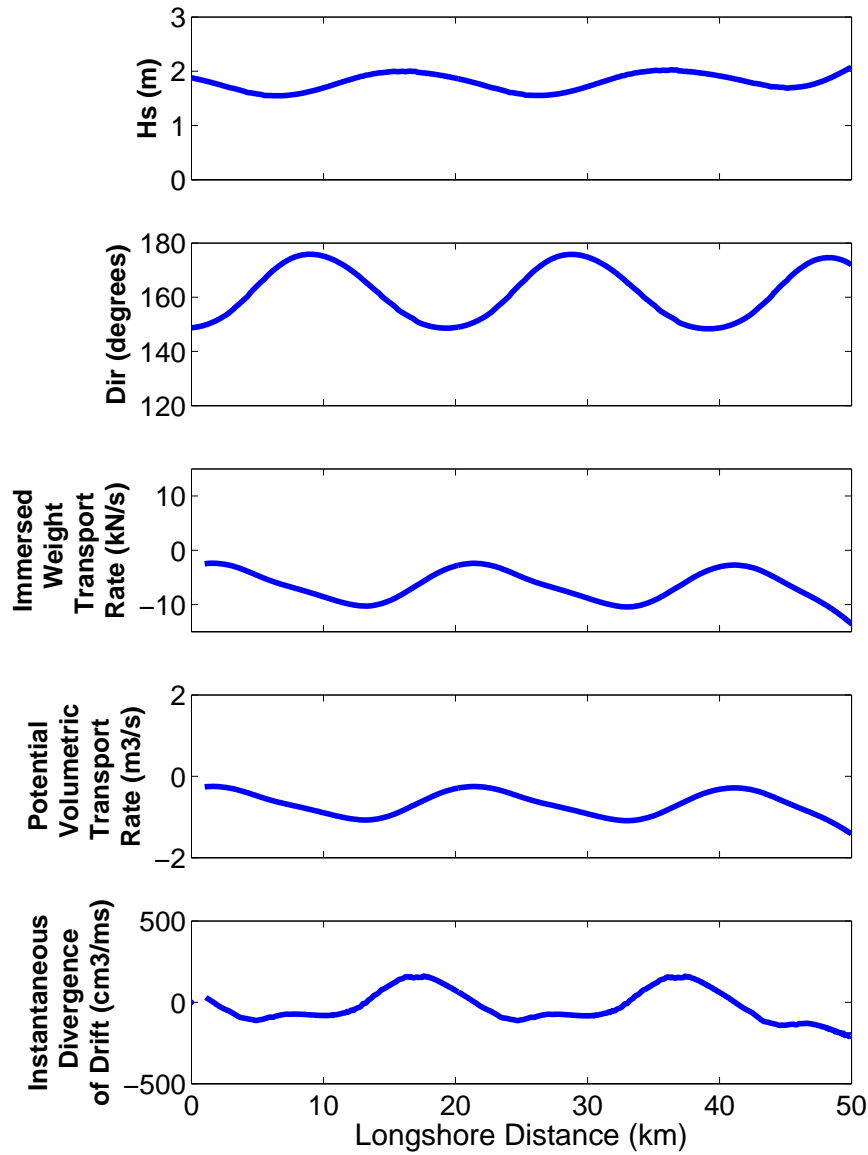


Figure A-6: The step wise calculation of wave height and modeled wave direction along the 5m isobath, and the calculation of longshore sediment transport rates and divergence of drift for the sinusoidal decaying continental shelf bathymetry (described in section A.2.2) using the input conditions of  $H_s=2\text{m}$ ,  $T=10\text{sec}$ , and  $Dir=135^\circ$ .

*last modified March 9, 2012*



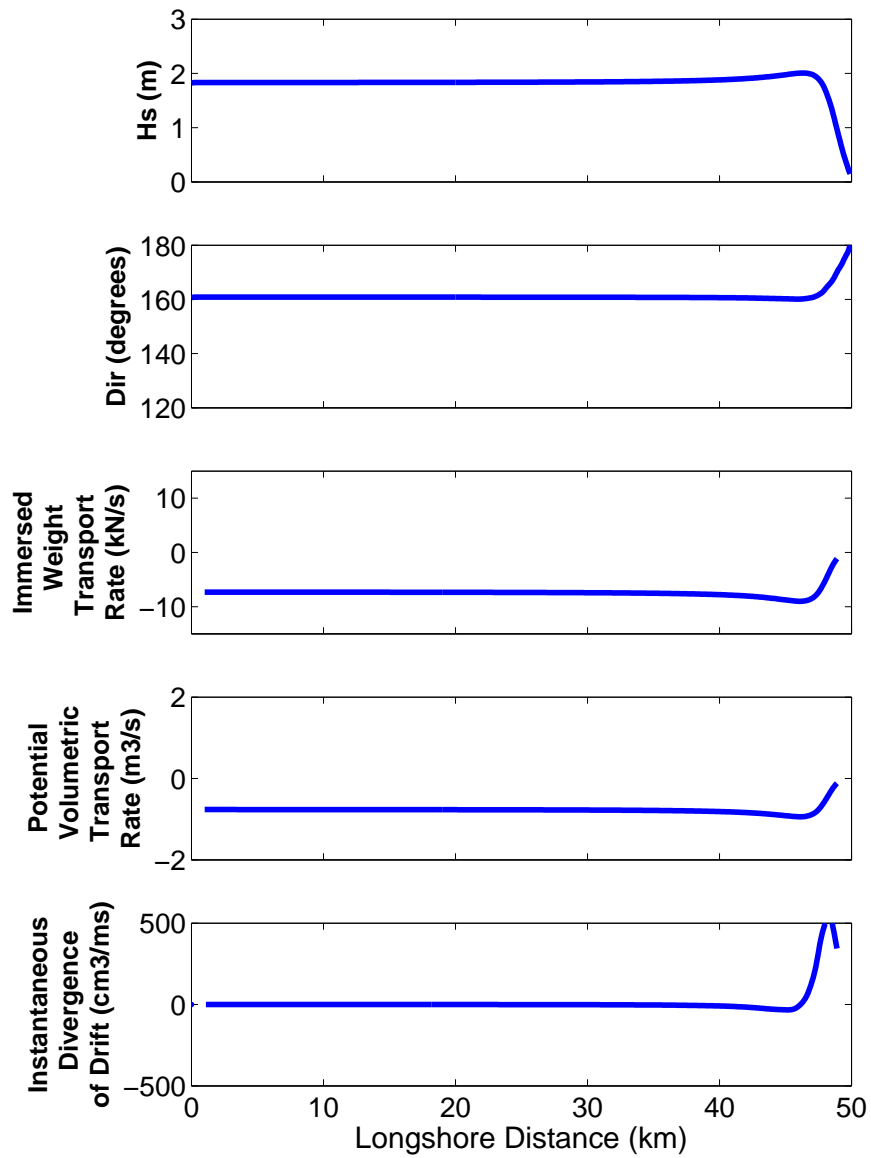


Figure A-7: The step wise calculation of wave height and modeled wave direction along the 5m isobath, and the calculation of longshore sediment transport rates and divergence of drift for the continental shelf bathymetry with an exponentially decaying slope (described in section A.2.3) using the input conditions of  $H_s=2\text{m}$ ,  $T=10\text{sec}$ , and  $\text{Dir}=135^\circ$ .

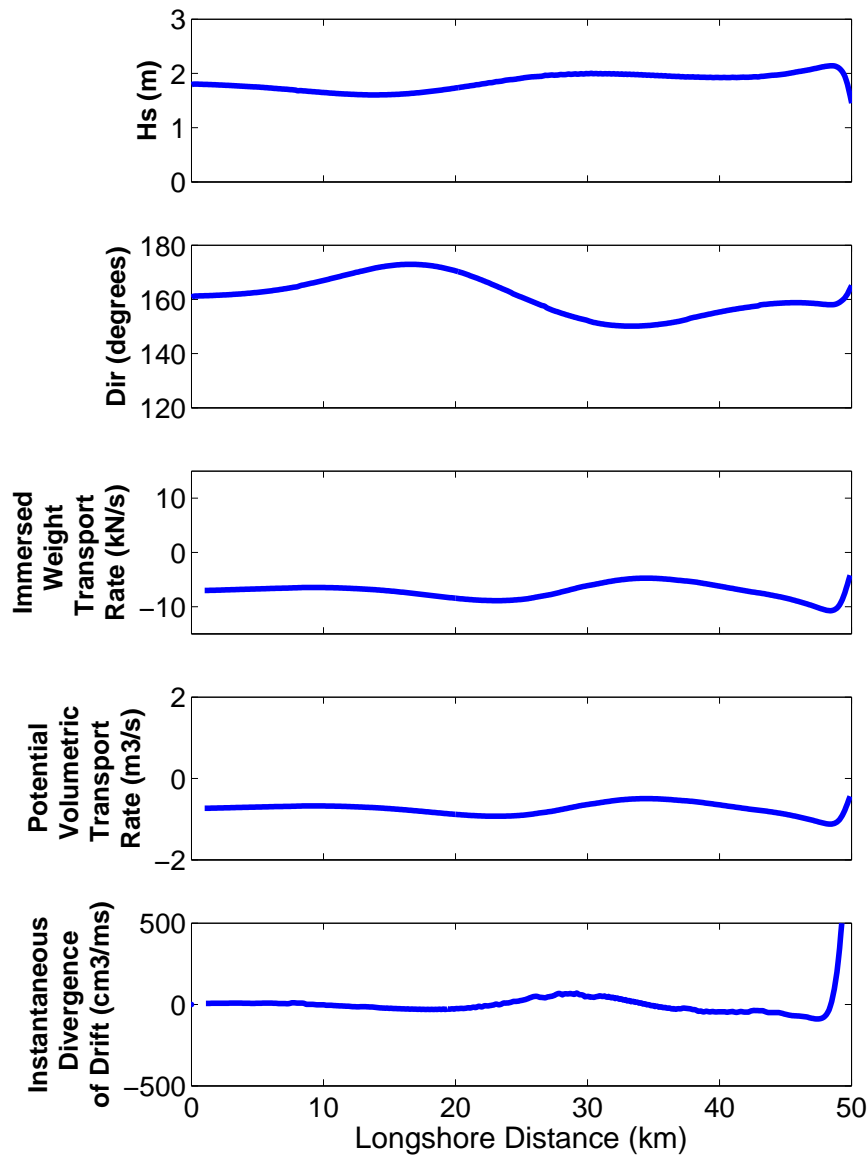


Figure A-8: The step wise calculation of wave height and modeled wave direction along the 5m isobath, and the calculation of longshore sediment transport rates and divergence of drift for the planar slope with a headland shaped continental shelf bathymetry (described in section A.2.4) using the input conditions of  $H_s=2\text{m}$ ,  $T=10\text{sec}$ , and  $\text{Dir}=135^\circ$ .

*last modified March 9, 2012*

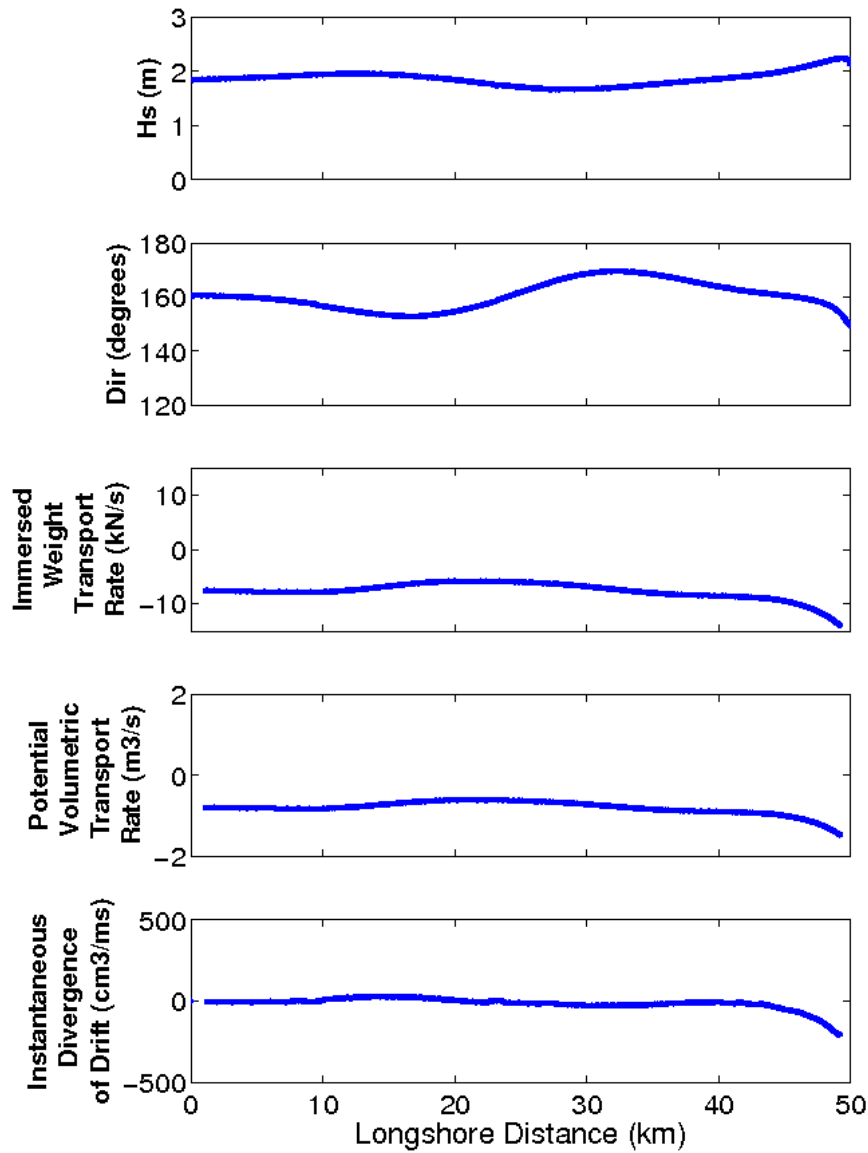


Figure A-9: The step wise calculation of wave height and modeled wave direction along the 5m isobath, and the calculation of longshore sediment transport rates and divergence of drift for the planar slope with a canyon shaped continental shelf bathymetry (described in section [A.2.5](#)) using the input conditions of  $H_s=2\text{m}$ ,  $T=10\text{sec}$ , and  $Dir=135^\circ$ .

*last modified March 9, 2012*

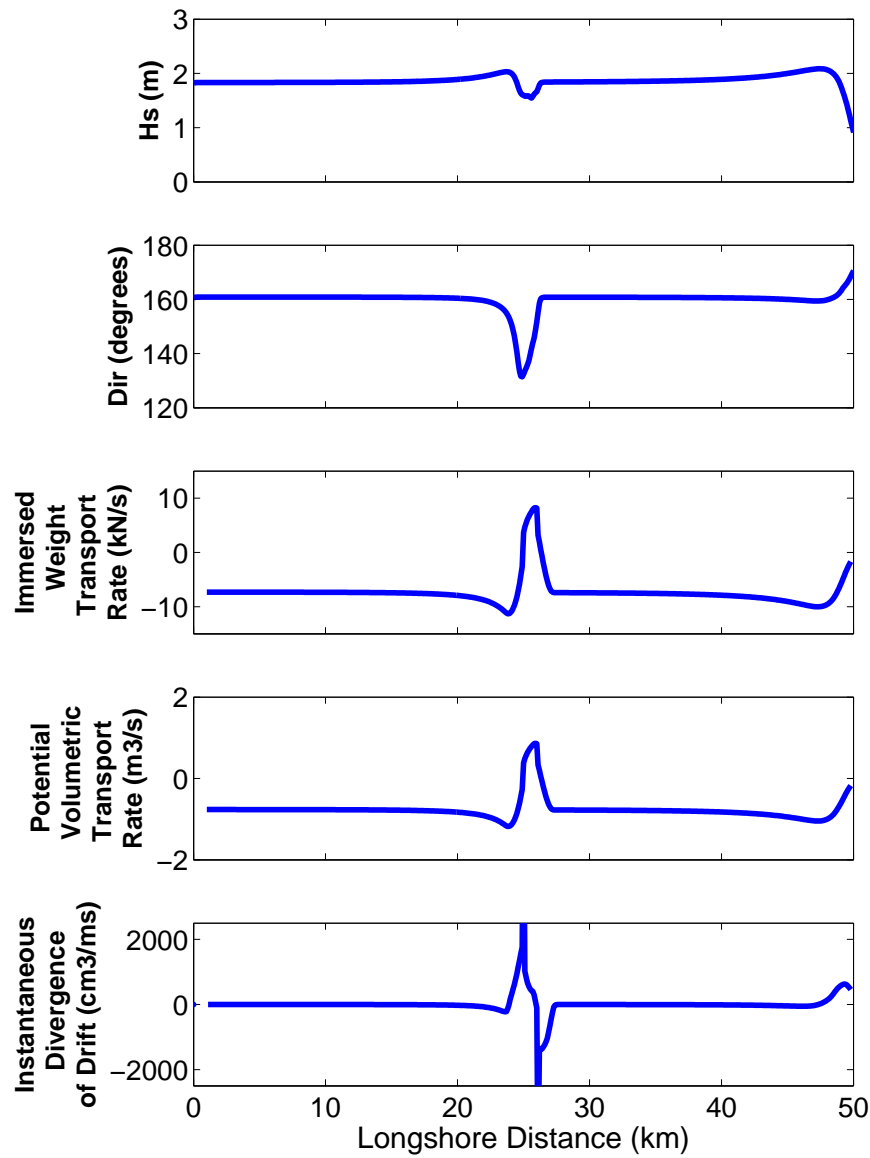


Figure A-10: The step wise calculation of wave height and modeled wave direction along the 5m isobath, and the calculation of longshore sediment transport rates and divergence of drift for the planar slope with hyperbolic tangent shaped coastline shaped continental shelf bathymetry (described in section A.2.6) using the input conditions of  $H_s=2\text{m}$ ,  $T=10\text{sec}$ , and  $Dir=135^\circ$ .

*last modified March 9, 2012*

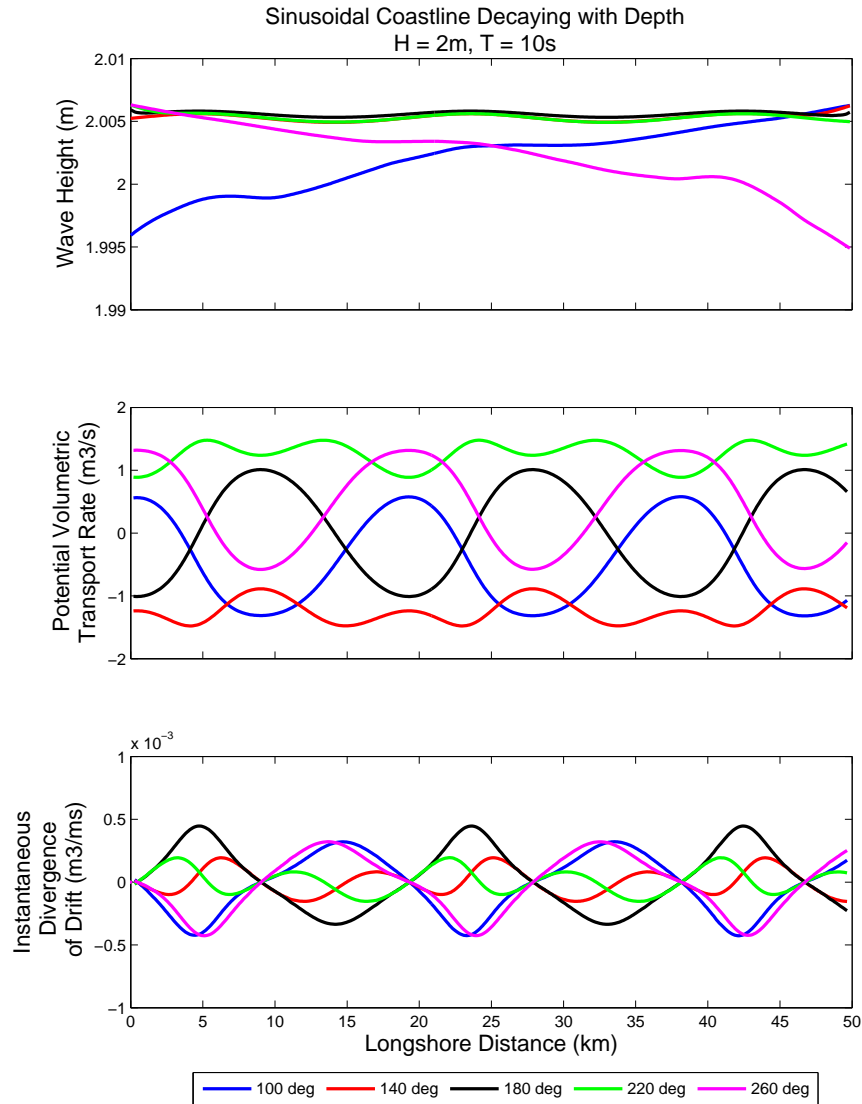


Figure A-11: The step wise calculation of wave height, longshore volumetric sediment transport rate and divergence of drift along the 5m isobath for the sinusoidal decaying continental shelf bathymetry (described in section A.2.2) using the input conditions of  $H_s=2\text{m}$ ,  $T=10\text{sec}$  with varying modelled wave directions shown.

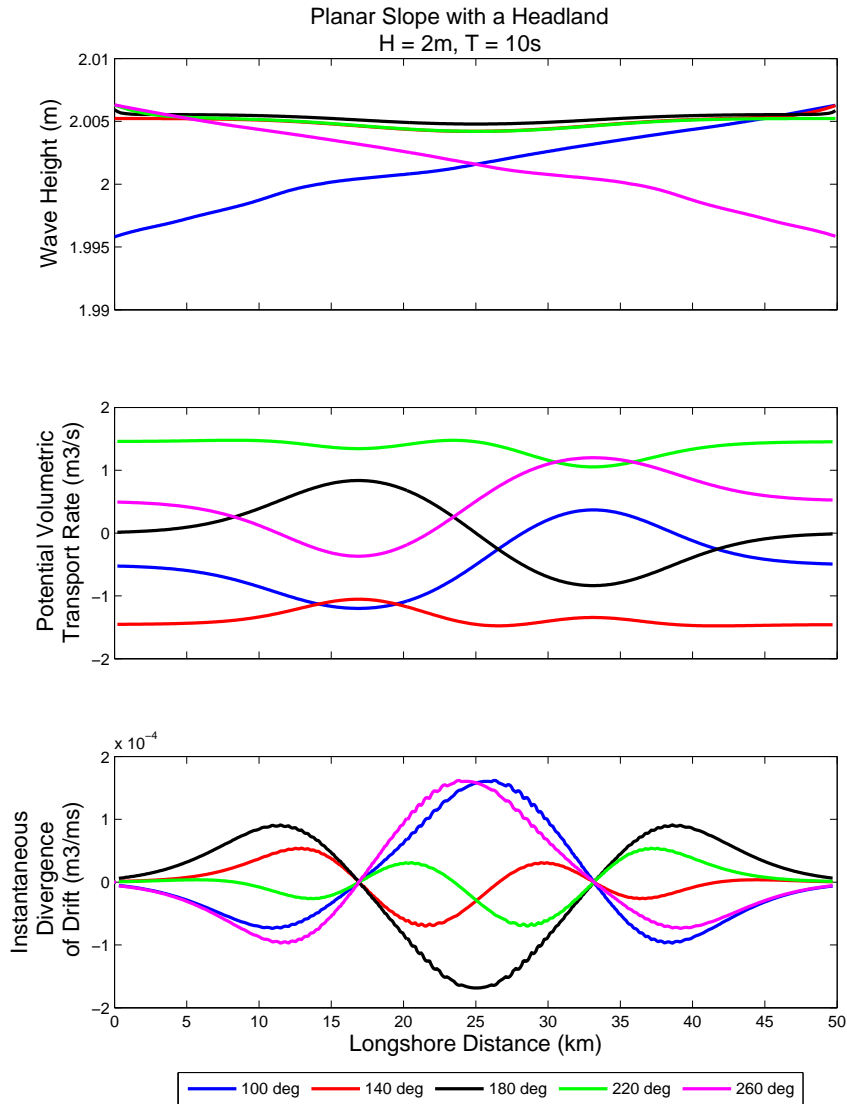


Figure A-12: The step wise calculation of wave height, longshore volumetric sediment transport rate and divergence of drift along the 5m isobath for the planar slope with a headland shaped continental shelf bathymetry (described in section A.2.4) using the input conditions of  $H_s=2\text{m}$ ,  $T=10\text{sec}$  with varying modelled wave directions shown.

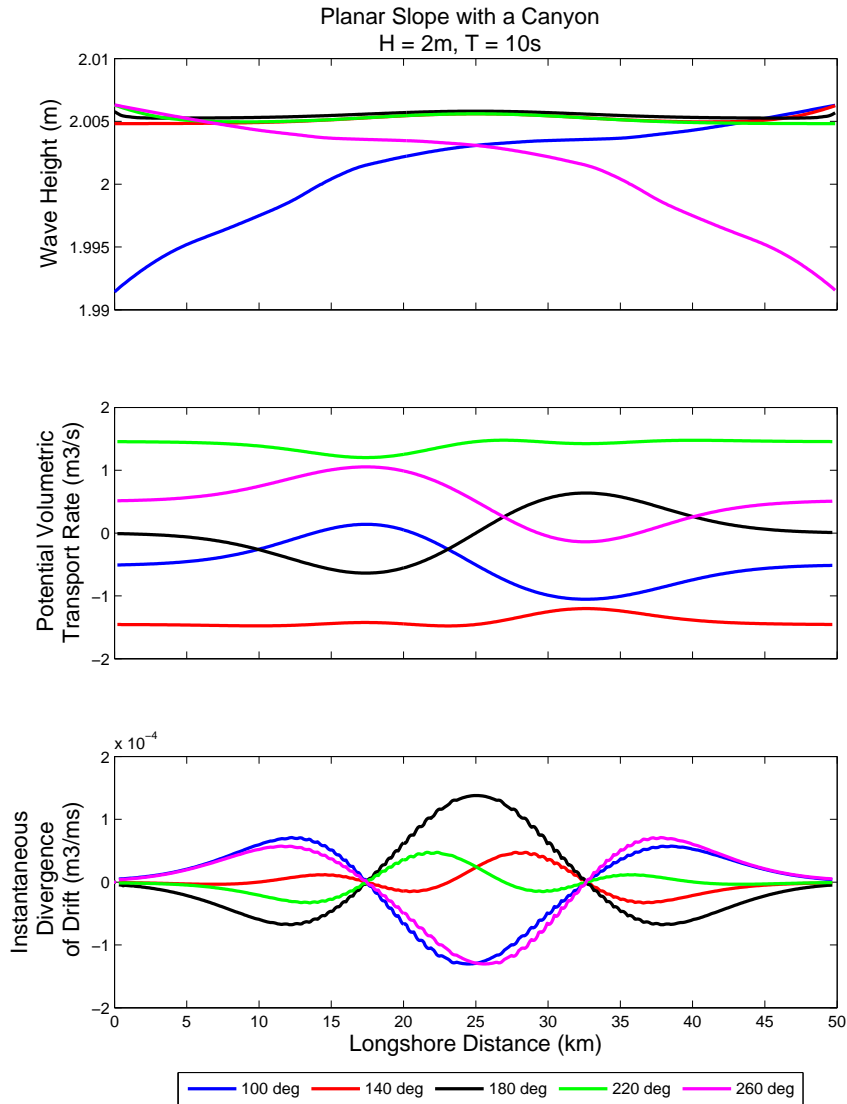


Figure A-13: The step wise calculation of wave height, longshore volumetric sediment transport rate and divergence of drift along the 5m isobath for the planar slope with a canyon shaped continental shelf bathymetry (described in section A.2.5) using the input conditions of  $H_s = 2\text{m}$ ,  $T = 10\text{s}$  with varying modelled wave directions shown.

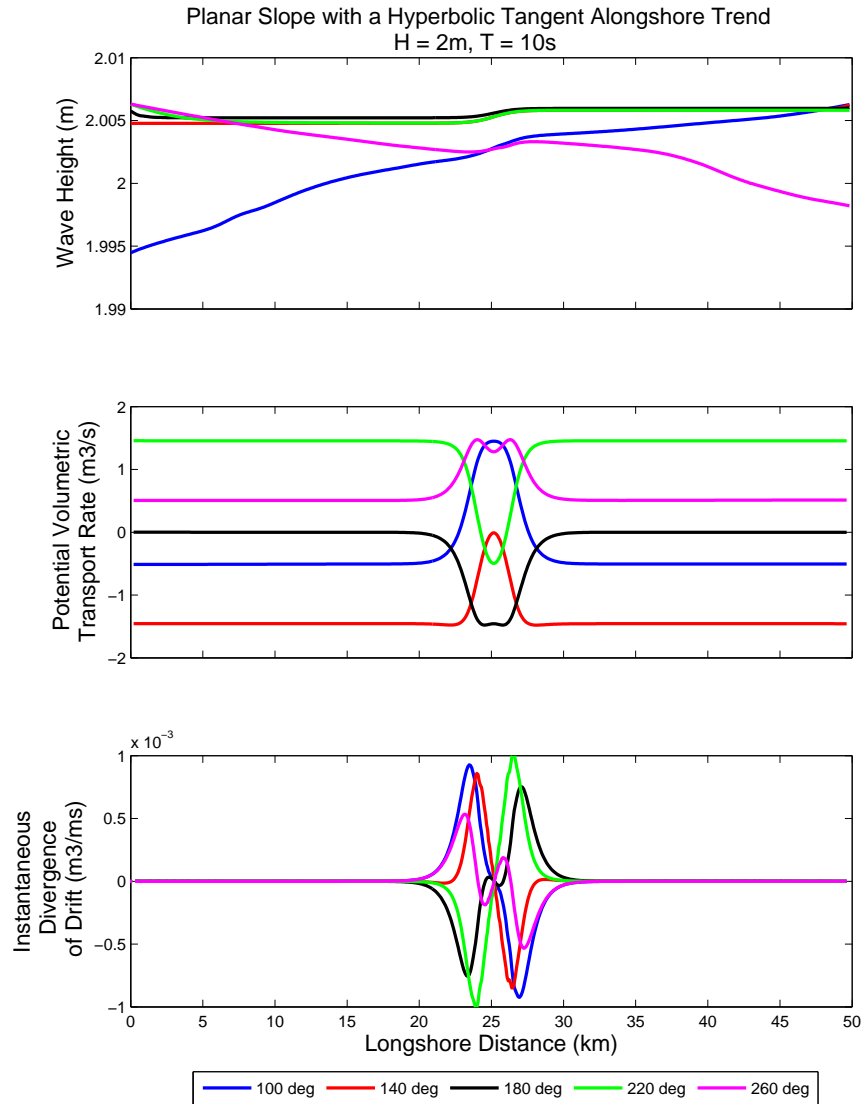


Figure A-14: The step wise calculation of wave height, longshore volumetric sediment transport rate and divergence of drift along the 5m isobath for the planar slope with hyperbolic tangent shaped coastline shaped continental shelf bathymetry (described in section A.2.6) using the input conditions of  $H_s=2\text{m}$ ,  $T=10\text{sec}$  with varying modelled wave directions shown.



## B An Example from the Southern California Bight

### B.1 Introduction

The Southern California Bight is a useful location to present how *WetSed* can be used on a complex bathymetry of an active margin coast. Through numerical experiments, the model was employed to illustrate its potential to explore impacts of climate change on decadal scale coastal evolution in this region.

### B.2 Regional Setting

For the purposes of this example, we consider the Southern California Bight to extend from a northwestern-most boundary at Point Arguello ( $34.58^\circ$ ,  $-120.65^\circ$ ) to the U.S.-Mexico border ( $32.54^\circ$ ,  $-117.12^\circ$ ) south of San Diego. Within this region, we have also established several subregions (10 km to 100 km reaches) where coastal evolution can be modeled and studied with higher spatial resolution outlined in figure B-1 referred to here as “nests.”

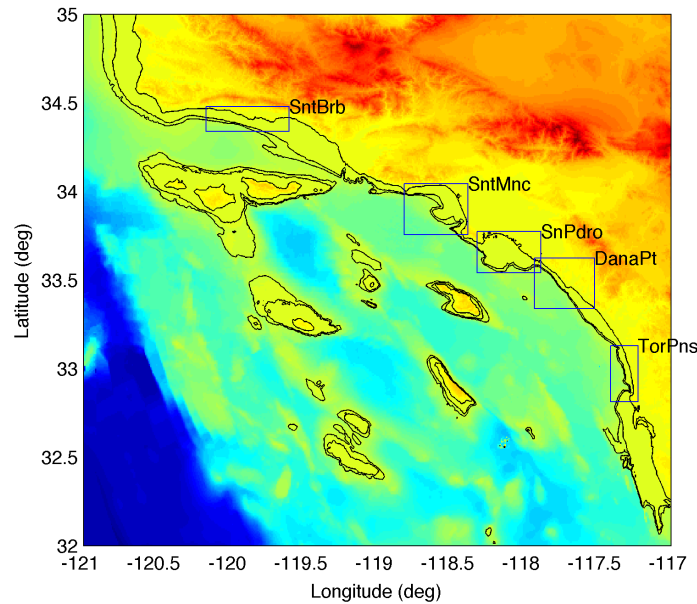


Figure B-1: Map of the Southern California Bight showing the five nested regions used in this example for detailed modeling analysis

*last modified March 9, 2012*

### B.3 Bathymetric Data

This example uses a grid of bathymetry of the Southern California Bight sea floor with a spatial resolution of 3 arc-seconds ( $\sim 93$  meters latitudinal spacing,  $\sim 77$  meters longitudinal spacing), ranging from  $32^\circ$  to  $35^\circ$  north latitude and  $-121^\circ$  to  $-117^\circ$  longitude ( $3600$  by  $4800 = 1.728 \times 10^7$  grid cells). Bathymetric data used here were obtained from the National Geophysical Data Center (NGDC-NOAA) 3 arc-second U.S. coastal relief model grid database.<sup>1</sup> This database provides coverage of nearshore, shelf, and proximal deep ocean bathymetry for the coterminous U.S. coastline, including Hawaii and Puerto Rico. By using a grid-based bathymetry, changing sea level is a trivial matter performed simply by adding a scalar value to each element of water depths in the bathymetric matrix. This is further simplified by the seamless coverage of the database from offshore to onshore terrain.

### B.4 Offshore Wave Climate

For this example a variety of wave fields are used to illustrate the results of this model, but hindcast information is available for numerous sites along the U.S. coast through the NOAA National Buoy Data Center (NDBC) website<sup>2</sup> and the Coastal Data Information Page (CDIP)<sup>3</sup>

---

<sup>1</sup>[www.ngdc.noaa.gov/mgg/coastal/coastal.html](http://www.ngdc.noaa.gov/mgg/coastal/coastal.html)

<sup>2</sup>[www.ndbc.noaa.gov/](http://www.ndbc.noaa.gov/)

<sup>3</sup><http://cdip.ucsd.edu/>

## B.5 Wave Transformation Results

### B.5.1 Coarse Resolutions Wave Field Computation

Figure B-2 provides examples of the SWAN computation for the entire Southern California Bight wave field, under various wave height and period conditions for a northwesterly wave direction. These simulations are run at the coarse resolution of 30 arc-seconds, or approximately 1 km grid spacing. Notice the prominent sheltering effects of the Channel Islands. Also apparent in these simulations is the profound effect of wave period on refraction (wave steering in shallow water).

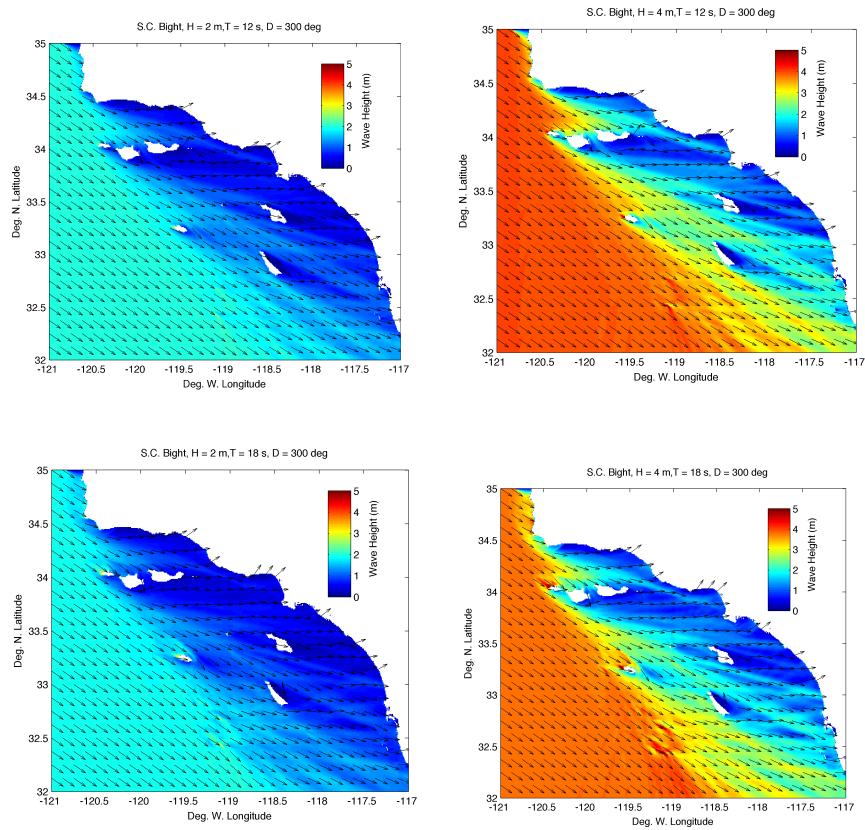


Figure B-2: Example SWAN runs showing wave height distribution over the entire Southern California Bight, for four separate sets of offshore wave conditions.

### B.5.2 Fine Resolutions Wave Field Computation

Figure B-3 provides examples of the SWAN computation for the Santa Barbara nested grid within the northern portion of the Southern California Bight wave field. These simulations are run at a fine resolution of 3 arc-seconds, or approximately 90-meter grid spacing. As in the case of the main grid above, arrows on the diagram illustrate wave ray direction, which further underscores the effect of wave period on the steering of wave rays due to refraction in shallow water.

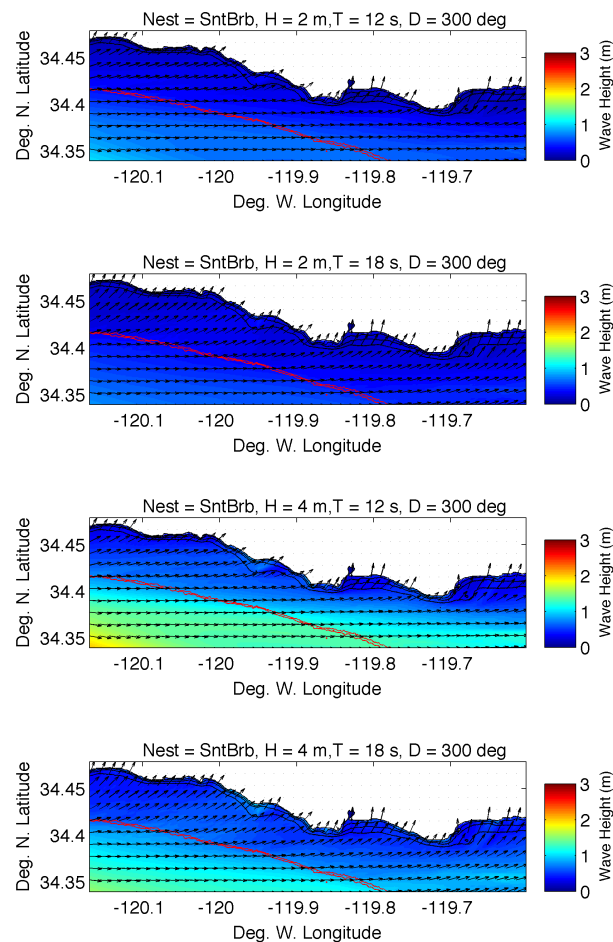


Figure B-3: Example SWAN runs showing wave height distribution over Santa Barbara newst within the northern portion of the Southern California Bight, for four separate sets of offshore wave conditions.

## B.6 Longshore Sediment Transport Model Output

Shown below is an example of stepwise calculations of wave energy flux for the SntMnc (location show in figure B-1) along the 5m isobath for the case of  $H=3\text{m}$ ,  $T=13\text{s}$ ,  $D=290^\circ$ . See section 1.4 for a more detailed explanation of the equations used in these calculations.

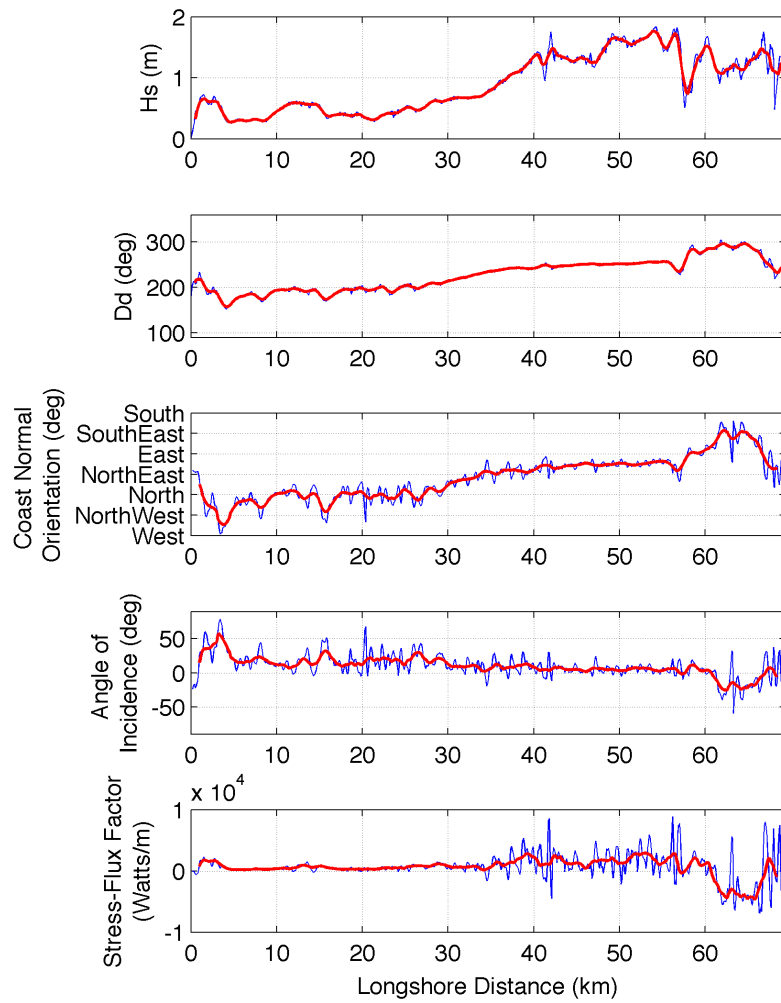


Figure B-4: Stepwise calculations of alongshore variability of modeled wave height, modeled wave direction, coast normal orientation, angle of incidence, and stress-flux factor for an example simulation in Santa Monica Bay.

The calculation of the volumetric rate of longshore sediment transport yields a rather noisy result. To obtain a more reasonable estimate of local trends in longshore sediment transport, we smooth the calculations of longshore transport with a 1-km moving average window .

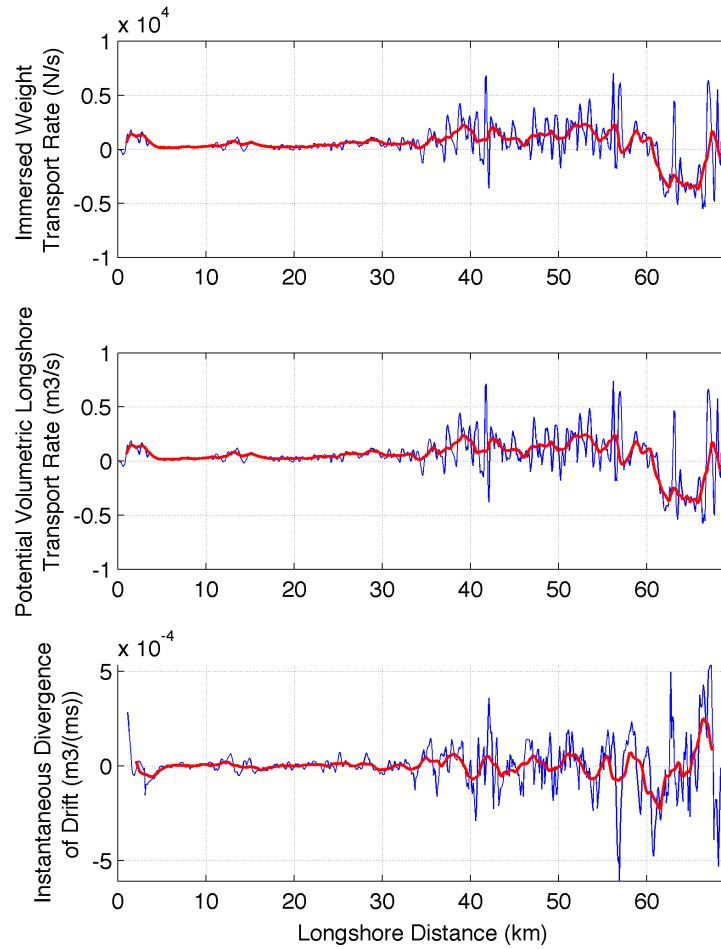


Figure B-5: Example calculation of longshore sediment transport rates and divergence of drift for Santa Monica Bay

## C Printed Matlab Codes

### C.1 wetsedExecuter.m

#### Contents

- Read Inputs
- Assign Directory Information
- Assign Nest Information based on 'prj\_s'
- Calls SwanControl.m

This is the main controlling script for running WetSed Created by J. Lovering 02/2009

```
disp(['Beginning WESTED run ' datestr(now)])
```

#### Read Inputs

```
[user prj_s projectName num_processes wetsed_ver ...
  Hs_range T_range Dir_range spread] = wetsedInputVariables;
pwd
```

#### Assign Directory Information

```
input_dir=['/Research/WETSED' wetsed_ver '/INPUTS/'];
base_dir=['/Research/WETSED' wetsed_ver '/OUTPUTS/' user '/' prj_s '/'];
bathy_names = { 'scb' 'nfa' 'pln' 'sns' 'xpd' 'hdl' ...
  'tnh' 'cyn' 'bch' };
```

#### Assign Nest Information based on 'prj\_s'

```
[alp mx my totcells xlen ylen xp yp nest_names ...
  bathy2run nests2run cgridscale] = wetsedNestDetails(prj_s);
```

#### Calls SwanControl.m

```
for jjj = bathy2run
  bathy_id = ['n' num2str(jjj)];
  prj_s = bathy_names{jjj};

  bathy_grd_name=['/Research/WETSED' wetsed_ver '/INPUTS/BathyGrids/' ...
    prj_s '_swan.txt'];

  [Hs,T,Dir] = swanControl(prj_s, Hs_range, T_range, Dir_range, spread,
    input_dir, base_dir, nests2run, bathy_grd_name, projectName,...
    num_processes,wetsed_ver,user,alp, mx, my, totcells, xlen, ylen,...
    xp, yp, nest_names,cgridscale);
end
```

*last modified March 9, 2012*

## C.2 wetsedInputVariables.m

### Contents

- Project Details
- Wave Field Conditions for SWAN boundary Conditions

```
function [user prj_s projectName num_processes wetsed_ver Hs_range...  
         T_range Dir_range spread] = wetsedInputVariables_v1
```

### Project Details

- The name of the user: (ie. kmalone, loving, adamsp, or manu) - critical for mpi use (swanControl.m automatically cd's to user's home directory to properly run mpi (parallelization))
- The three letter code to represent the name of the project (ie. scb, nfa) or for simple geometric grids use the codes pln, sns to run individuals or use and sgg to run them all
- The project name to be placed on the top of the SWAN input file
- The number of Processes used for MPI (ie. 20)

```
disp('running wetsedInputVariables.m')  
pwd  
user = 'loving';  
prj_s = 'pln';  
projectName = ' '  
num_processes = 16;
```

```
% IF MPI is not working:  
%num_processes = 1;
```

```
wetsed_ver='1.2';
```

### Wave Field Conditions for SWAN boundary Conditions

- Input Wave Height ranges in (1:1:10) format or as a single value
- Input Wave Period ranges in (1:1:10) format or as a single value
- Input Wave Direction ranges in (1:1:10) format or as a single value
- Input the coefficient of directional spread

```
Hs_range=[1.5:0.1:6];  
T_range=[8:0.5:18];  
Dir_range=[180 210 240];  
spread=15;
```

```
disp('complete wetsedInputVariables.m')
```



### C.3 wetsedNestDetails.m

This file will need to be manually updated by user, it will need to include the grid details for each project based on their project string ID ('prj\_s'), two examples are given below. The first example has two nested grids, and the second does not have any. Variable descriptions given in manual.

```
function [alp mx my totcells xlen ylen xp yp nest_names...
    bathy2run nests2run cgridscale] = wetsedNestDetails(prj_s)

disp('running wetsedNestDetails.m')
pwd

if (strcmp (prj_s,'nfa'))
    cgridscale = 10;
    bathy2run = [2];
    nests2run = [ 1 2 ];
    nest_names = { 'CapCnv' 'MtzInt' };
    alp = [0 0];
    mx = [450 550];
    my = [ 600 600];
    totcells = [27000 330000];
    xlen = [ 35928.315 39920.35];
    ylen = [55555.56 55555.56];
    xp = [ 123753.085 63872.56];
    yp = [27777.78 173148.162];

elseif (strcmp (prj_s,'sgg'))
    cgridscale = 1;
    bathy2run = [3 4 5 6 7 8];
    nests2run = [ ];
    nest_names = { };
    alp = [ ];
    mx = [ ];
    my = [ ];
    totcells = [ ];
    xlen = [ ];
    ylen = [ ];
    xp = [ ];
    yp = [ ];

end
disp('complete wetsedNestDetails.m')

end
```

*last modified March 9, 2012*

## C.4 swanControl.m

### Contents

- INPUTS (can be modified by user here)
- Retrieves Computations Grid data Based on prj\_s
- Write name of bathymetry input file
- Begin MPI usage
- Constructs a synthetic time series of Deep Water Wave Conditions
- Creates Output file Directories for each set of H,T,Dir combination in /SwanOutput/
- MAIN GRID
- NESTED GRIDS
- Makes Output Graphs
- Clean up extra files & Stop Multiprocessing

```
function[Hs,T,Dir]=swanControl(prj_s, Hs_range, T_range, Dir_range, spread,...
    input_dir, base_dir, nests2run, bathy_grd_name,projectName,...
    num_processes,wetsed_ver,user,alp, mx, my, totcells, xlen,...
    ylen, xp, yp, nest_names,cgridscale)
```

Control file for SWAN simulations. This m-file runs the main simulation, plus nested runs.

Originally inspired by a meeting with Kraig Winter at SIO (March 2006) created by PNA at UF, Jan. 30, 2007 mod. to loop thru storm event, hour-by-hour by PNA at UF, Feb.9, 2007 mod. to run in parallel on cluster by Manu and PNA at UF, May 5, 2008

```
disp('running swanControl.m')
pwd
```

### INPUTS (can be modified by user here)

```
batch_num='005';
first_runnum=101849;
```

### Retrieves Computations Grid data Based on prj\_s

```
[dxinp,dyinp, num_cols, num_rows] = swanPrjDetails(prj_s,input_dir);
pwd
```

### Write name of bathymetry input file

```
bathy_grd_name=['/Research/WETSED' wetsed_ver '/INPUTS/BathyGrids/' prj_s '_swan.txt'];
```

*last modified March 9, 2012*

## Begin MPI usage

```

if (strcmp (user,'adamsp'))
    old_dir=pwd
    eval(['cd /home/' user '/''])
    ! mpdboot -n 4
    cd(old_dir)
else
    old_dir=pwd;
    eval(['cd /home/' user '/'']);
    ! mpdboot -n 5
    cd(old_dir);
end

```

## Constructs a synthetic time series of Deep Water Wave Conditions

- Calls the function *swanDwvmaker.m* if wave conditions are specified for a range of values
- A mat file is created and loaded with values of Hs\_all, T\_all, Dir\_all, and run\_nums unless only one values

```

LOOP = length(Hs_range)*length(T_range)*length(Dir_range);

```

```

if (LOOP ~= 1)

    [run_nums,Hs_all,T_all,Dir_all]=swanDwvmaker(input_dir,batch_num,...
        first_runnum,Hs_range,T_range,Dir_range);
    load([input_dir 'WvInput_b' batch_num '.mat']);
else
    Hs_all = Hs_range;
    T_all = T_range;
    Dir_all = Dir_range;
    disp('(skipped Dwvmker)')
    run_nums = first_runnum;
end

```

## Creates Output file Directories for each set of H,T,Dir combination in /SwanOutput/

- This is the Start of the main loop for each set of wave conditions

```

for iii=1:length(Hs_all)

    run_num_s=num2str(run_nums(iii));
    Hs=Hs_all(iii);T=T_all(iii);Dir=Dir_all(iii);
    new_dir=sprintf('%s%s_H%04.1f_T%04.1f_D%03i',base_dir,prj_s,Hs,T,Dir) ;
    mkdir(new_dir);
    current_dir=pwd;
    cd(new_dir);

```

*last modified March 9, 2012*

## MAIN GRID

- Call the function *swanInputWriter.m* to build the input for for the main grid run
- Call the function *swanExecuter.m* to begin SWAN run
- Records the time to run main grid in variable "main\_time"

```

nest_id = 'main';

%      [bathy_grd_params_s] = swanInputWriter(run_num_s,Hs,T,Dir,...
%      nest_id,spread,num_rows,num_cols,dyinp,dxinp);

[bathy_grd_params_s]=swanInputWriter(run_num_s,Hs,T,Dir,...
    nest_id,spread,num_rows,num_cols,dyinp,dxinp,bathy_grd_name,prj_s,new_dir,...
    mx,my,nest_names,xp,yp,xlen,ylen,alp,projectName,user,wetsed_ver,cgridscale);

tic
%      swanExecuter(run_num_s,nest_id);
pwd

swanExecuter(run_num_s, nest_id, num_processes, prj_s)

    nest_strtemp = prj_s;
if (strcmp (prj_s,'scb'))
    progress = 'skipped'
elseif(strcmp (prj_s,'nfa'))
    progress = 'skipped'
else
    cgemAnalyze(prj_s,wetsed_ver,nest_strtemp,Hs,T,Dir,user)
end

main_time = toc;

```

## NESTED GRIDS

- Call the function *swanNestWriter.m* to write the nest input files
- Call the function *swanExecuter.m* to begin each nested SWAN run for nested grids
- Records the time to run nested grid in variable array "nest\_time"
- Will run for nested grids listed 'nests2run'

```

count = 1;
for jjj = nests2run

    nest_id = ['n' num2str(jjj)];
    nest_str = nest_names{jjj};

    swanNestWriter(run_num_s,nest_id,xp(jjj),yp(jjj),alp(jjj),...
        xlen(jjj),ylen(jjj),mx(jjj),my(jjj),nest_str,...
        bathy_grd_params_s,Hs,T,Dir,spread,...

```

*last modified March 9, 2012*

```

        bathy_grd_name, prj_s,new_dir,projectName);

tic

swanExecuter(run_num_s, nest_id, num_processes, prj_s)
nest_time(count) = toc;
count = count+1;

%         nest_str = prj_s;

%   cgemAnalyze(prj_s,wetsed_ver,nest_str,Hs_range,T_range,Dir_range)
cgemAnalyze(prj_s,wetsed_ver,nest_str,Hs,T,Dir,user)
end

```

## Makes Output Graphs

```

%   cd(new_dir);
%   swanOutputPlots(prj_s,new_dir)

```

## Clean up extra files & Stop Multiprocessing

```

! rm *_f *.prt*
! rm *.prt*
cd(current_dir);

end

! mpdallexit
disp('complete swanControl.m')

```

## C.5 swanPrjDetails.m

This function was created to load the correct grid dimensions according to the prj\_s (ie. scb, nfa) from mat files in /SwanInput/

Created by Jessica Lovering Feb 27, 2009 at UF

June 09, 2009 modified to handle both arc second spacing and regular grid spacing

```

function [dxinp,dyinp, num_cols, num_rows] = swanPrjDetails(prj_s,input_dir)

disp('running swanPrjDetails.m')
pwd
prj_filename = sprintf('%sGridVars/%s_grid_vars.mat',input_dir,prj_s);

load([prj_filename]);

if (strcmp (gridspacing,'cartesian'))

```

*last modified March 9, 2012*

```
elseif(strcmp(gridspacing, 'arcsec'))

    dyinp=(40e6/360)*(res_in_arcsec/3600);
    dxinp=(res_in_arcsec/3600)*...
    (2*pi*6371e3*sin((pi/180)*(90-abs(mid_lat)))/360);

else
    disp('Did not specify the grid spacing type in grid variable mat file')
end

disp('complete swanPrjDetails.m')
```

## C.6 swanDwwvmaker.m

Called in *swanControl.m*

```
function [run_nums,Hs_all,T_all,Dir_all] =
    swanDwwvmaker(input_dir,batch_num,first_runnum,H,T,Dir)
```

swanDwwvmaker.m (swan deep-water wave maker) - a file to construct a synthetic time series of deep water wave conditions (Hs, T, Direction), specified by the user.

- written by Manu Sethi, early May 2008, with modifications by PNA\*

```
disp('running swanDwwvmaker.m')
initCT = [] ;

runNum = first_runnum ;

for h = 1 : size (H,2)
    for t = 1 : size (T,2)
        for a = 1 : size (Dir,2)

            temp = [runNum , H(h) , T(t) , Dir(a)] ;
            initCT = [initCT ; temp] ;
            runNum = runNum + 1 ;

        end
    end
end
run_nums=initCT(:,1);
Hs_all=initCT(:,2);
T_all=initCT(:,3);
Dir_all=initCT(:,4);

save ([input_dir 'WvInput_b' batch_num '.txt'], 'initCT' , '-ascii') ;
save ([input_dir 'WvInput_b' batch_num '.mat'], ...
    'run_nums','Hs_all','T_all','Dir_all') ;
```

*last modified March 9, 2012*

## C.7 swanInputWriter.m

Called in *swanControl.m*

### Contents

- Calculate Grid Height and Width
- Define values to be used with computational grid (CGRID REGular and CIRcle statements)
- Define values to be used with bathymetric grid (INPgrid REGular statements)
- Define values to be used with boundary conditions (BOUNdspec SEG UNIFORM PAR commands)
- Define output grid information
- Write header information
- Write simulation conditions
- Write lines in main .swn input file to create nested grids (with a loop)
- Close the input file

```
function [bathy_grd_params_s]=swanInputWriter(run_num_s,Hs,T,Dir,...
    nest_id,spread,num_rows,num_cols,dyinp,dxinp,bathy_grd_name,prj_s,new_dir,...
    mx,my,nest_names,xp,yp,xlen,ylen,alp,projectName,user,wetsed_ver,cgridscale)
```

this function builds the main swan input file (.swn) to be run over a large region at coarse resolution - then it saves nest information to be used as boundary conditions for running the nested simulations that are built by 'swanNestWriter.m'

**This file was started by PNA at UF, Jan. 26, 2007**

... then things started clicking Jan. 29, 2007 successfully run with call to execute SWAN on ... Jan. 30, 2007 significantly modified to include looped-creation of nests Feb. 23, 2007

```
disp('running swanInputWriter.m')
%
```

### Calculate Grid Height and Width

ADDED x10 here

```
grid_height_m=(num_rows-1)*dyinp;
grid_width_m=(num_cols-1)*dxinp;
```

*last modified March 9, 2012*

## Define values to be used with computational grid (CGRID REGular and CIRcle statements)

```
xpc=0; % geogrphc x-loc. of origin of compgrid (in m)
ypc=0; % geogrphc y-loc. of origin of compgrid (in m)
alpc=0; % direc. of pos. x-axis of compgrid (in degrees)
xlenc=grid_width_m; % length of x-direc. of comp. grid (in m)
ylenc=grid_height_m; % length of y-direc. of comp. grid (in m)
mxc=floor(num_cols/cgridscale); % no. meshes in x-dir. comp. grid (no.grd.pts.-1)
myc=floor(num_rows/cgridscale); % no. meshes in y-dir. comp. grid (no.grd.pts.-1)
c_grd_params_s=[num2str(xpc) ' ' num2str(ypc) ' ' num2str(alpc) ' ' ...
    num2str(xlenc) ' ' num2str(ylenc) ' ' num2str(mxc) ' ' num2str(myc)];
mdc=36; % num. of meshes in theta space (delta theta =360deg/mdc)
flow=0.05; % lowest discrete frequency (in Hz)
fhigh=1; % highest discrete frequency (in Hz)
direc_params_s=[num2str(mdc) ' ' num2str(flow) ' ' num2str(fhigh)];
```

## Define values to be used with bathymetric grid (INPgrid REGular statements)

```
xpinp=0; % geogrphc x-loc. of origin of input (bathy) grid (in m)
ypinp=0; % geogrphc y-loc. of origin of input (bathy) grid (in m)
alpinp=0; % direc. of pos. x-axis of input (bathy) grid (in degrees)
mxinp=(num_cols-1); % num. meshes x-dir. of inp (bathy) grid(no.grd.pts.-1)
myinp=(num_rows-1); % num. meshes y-dir. of inp (bathy) grid(no.grd.pts.-1)
% dxinp defined above; % mesh size in x-dir. of input (bathy) grid (in m)
% dyinp defined above; % mesh size in y-dir. of input (bathy) grid (in m)
% dxinp and dyinp are now determined above in the call to grd_spc_calc
% bathy_grd_params=[xpinp ypinp alpinp mxinp myinp dxinp dyinp];
bathy_grd_params_s=[num2str(xpinp) ' ' num2str(ypinp) ' ' ...
    num2str(alpinp) ' ' num2str(mxinp) ' ' num2str(myinp) ' ' ...
    num2str(dxinp) ' ' num2str(dyinp)];
```

## Define values to be used with boundary conditions (BOUNDspec SEG UNIFORM PAR commands)

Correct i,j values for scb grid follow below:

```
i_vals=[20 0 0 0 0 floor(mxc*0.9)]; % defining the i coords of bound. segs.
j_vals=[myc myc myc 0 0 0]; % defining the j coords of bound. segs.
* Without clipping nfa grid so incorrect for nfa below
i_vals=[158 360 360 360 250]; % defining the i coords of bound. segs.
* Correct after clipping nfa for clipped nfa grid:
i_vals=[168 281 281 281 250]; % defining the clipped_nfa i coords of bound. segs.
j_vals=[600 600 600 0 0 0]; % defining the j coords of bound. segs.
above two lines are correct for nfa clipped grid but commented to use
function instead:
```

```
[i_vals j_vals] = swanCalcIJVals(prj_s,wetsed_ver,mxc,myc) ;
```

*last modified March 9, 2012*



```

i_j_vals_s=[num2str(i_vals(1)) ' ' num2str(j_vals(1)) ' ' ...
            num2str(i_vals(2)) ' ' num2str(j_vals(2)) ' ' ...
            num2str(i_vals(3)) ' ' num2str(j_vals(3)) ' ' ...
            num2str(i_vals(4)) ' ' num2str(j_vals(4)) ' ' ...
            num2str(i_vals(5)) ' ' num2str(j_vals(5)) ' ' ...
            num2str(i_vals(6)) ' ' num2str(j_vals(6))];

params_s=[num2str(Hs) ' ' num2str(T) ' ' num2str(Dir) ' ' num2str(spread)];

```

## Define output grid information

```

ix1=0; ix2=mx; iy1=0; iy2=my;
i_j_outs_s=[num2str(ix1) ' ' num2str(ix2) ' ' num2str(iy1) ' ' ...
            num2str(iy2)];

%
run_filename=[new_dir '/swan_' prj_s '_' run_num_s nest_id '.swn'];
%
% Build the input file (.swn) and save it in the appropriate place
%
% name & open the main run .swn file
fid=fopen(run_filename,'w');
%

```

## Write header information

```

fprintf(fid,'%s\n',['! PROJECT ' projectName ', Run ID Number: '...
                run_num_s nest_id]);
fprintf(fid,'%s\n',['! Run Clock: ' datestr(now)]);
fprintf(fid,'%s\n',['! USER: ' user]);
fprintf(fid,'%s\n',['! wave conditions: ' num2str(Hs) ' m, ' num2str(T) ...
                ' s, ' num2str(Dir) ' deg, (' num2str(spread) ' deg spread)']);
fprintf(fid,'%s\n',['! ']);
fprintf(fid,'%s\n',['! ']);

```

## Write simulation conditions

```

fprintf(fid,'%s\n',['SET NAUTical']);
fprintf(fid,'%s\n',['MODE STATIONARY TWODimensional']);
fprintf(fid,'%s\n',['COORDinates CARTesian']);
fprintf(fid,'%s\n',['CGRID REGular ' c_grd_params_s ' CIRCLE ' ...
                direc_params_s]);
fprintf(fid,'%s\n',['INPgrid BOTtom REGular ' bathy_grd_params_s ...
                ' EXC -99999']);
fprintf(fid,'%s\n',['READinp BOTtom 1 ' ' ' bathy_grd_name ' ' ' ' 1 0 FREE']);
fprintf(fid,'%s\n',['BOUND SHAPespec JONswap 3.3 PEAK DSPR DEGR']);
fprintf(fid,'%s\n',['BOUNDspec SEG IJ ' i_j_vals_s ' UNIFORM PAR ' params_s]);
fprintf(fid,'%s\n',['OFF QUAD']);

```

*last modified March 9, 2012*

```
fprintf(fid,'%s\n',['GROUP ''compgrid'' SUBGRID ' i_j_outs_s]);
```

## Write lines in main .swn input file to create nested grids (with a loop)

```
for i=1:length(mx)
    fprintf(fid,'%s\n',['NGRID ''' nest_names{i} '_s'' ' ...
        num2str(xp(i)) ' ' num2str(yp(i)) ' ' num2str(alp(i)) ' ' ...
        num2str(xlen(i)) ' ' num2str(ylen(i)) ' ' num2str(mx(i)) ' ...
        ' num2str(my(i))]);
end
%fprintf(fid,'%s\n',['BLOCK ''compgrid'' NOHEADER ''out_grid.mat'' HSIGN...
    DIR XP YP DEPTH TMM10 QB']);
%fprintf(fid,'%s\n',['BLOCK ''compgrid'' NOHEADER ''out_' prj_s '_grid.mat''...
    HSIGN DIR XP YP DEPTH']);
fprintf(fid,'%s\n',['BLOCK ''compgrid'' NOHEADER ''out_' prj_s '_grid.mat''...
    HSIGN DIR XP YP DEPTH QB VEL TRANSP FORCE']);
% write lines in main .swn input file to do the NESTout cmd. (with a loop)
for i=1:length(mx)
    fprintf(fid,'%s\n',['NEStout ''' nest_names{i} '_s'' ' '''' nest_names{i}...
        '_f'' ']);
end
fprintf(fid,'%s\n',['COMPUTE']);
fprintf(fid,'%s\n',['STOP']);
```

## Close the input file

```
fclose(fid);
```

## C.8 cgemAnalyze.m

### Contents

- Enter project name, directory information, wave coeff, and grid parameters
- Load isobath and decimate
- Measure coastal trends along decimated 5m isobath
- Retrieve swan results for Nest
- Interpolate swan results along decimated 5m isobath
- Calculate angle of incidence along 5m isobath
- Compute WvEnrgy, Wv.EnrgyFlx, Sxy, and StrssFlxFactor
- Compute divergence of drift along 5m isobath

### Code Description

Code to identify coastal hotspots of erosion for proscribed time series of deep-water wave field (Hs,T,Dir).

Code can be looped by executing the following line at the command prompt:

*last modified March 9, 2012*

by pna at UF, late May-early June, 2008 modified by pna in flight, Aug. 1, 2008 modified by jll to accomodate simple bathymetry runs June 5, 2009

```
function cgemAnalyze(prj_s,wetsed_ver,nest_str,H,T,Dir,user)

disp('running cgemAnalyze.m')
```

## 1. Enter prject name, directory information, wave coeff, and grid parameters

```
% Directory Assignment
input_dir=['/Research/WETSED' wetsed_ver '/INPUTS/'];
output_dir = sprintf(['/Research/WETSED' wetsed_ver '/OUTPUTS/' user '/' ...
    prj_s '/%s_H%04.1f_T%04.1f_D%03i/'],prj_s,H,T,Dir) ;
isobaths_dir=['/Research/WETSED' wetsed_ver '/INPUTS/5mIsobathLoc'];
results_dir=output_dir;
if exist(results_dir)~=7,mkdir(results_dir); end

% Grid Parameters
sm_win_sz=10;
coast_sm_win_sz1=5;
coast_sm_win_sz2=20;
dec_spc=100;

K=0.8;

disp('Done - cgemAnalyze Setup')
```

## 2. Load isobath and decimate

```
load([isobaths_dir '/' 'Isobath_' nest_str '.mat']);
[Dist_05]=[0; cumsum(sqrt((diff(x_05).^2)+(diff(y_05).^2)))];
[Dist_05_i,x_05_i,y_05_i]=cgemDecimateIsobath(Dist_05,x_05,y_05,dec_spc);
LP_05_i=1:1:length(x_05_i);
disp('Done - cgemDecimateIsobath')
plot(x_05_i,y_05_i)
```

## 3. Measure coastal trends along decimated 5m isobath

```
[CT_05_i,CT_u_05_i,Q,CN_05_i,m_h,m_v,range_x,range_y]=...
    cgemComputeCsttrnd(x_05_i,y_05_i,coast_sm_win_sz1,prj_s);

[CTsm,CT_u_sm,Q,CNsm,mhsm,mvsm,rangexsm,rangeysm]=...
    cgemComputeCsttrnd(x_05_i,y_05_i,coast_sm_win_sz2,prj_s);
disp('Done - cgemComputeCsttrnd')
```

*last modified March 9, 2012*

#### 4. retrieve swan results for Nest

- Loads out\_grid.mat to get variables Xp, Yp, Hsig, Depth, and Dir

```
i=1;

cdDir = sprintf(['/Research/WETSED' wetsed_ver '/OUTPUTS/' user '/' prj_s...
               '/%s_H%04.1f_T%04.1f_D%03i/'],prj_s,H,T,Dir) ;
readFile = sprintf ('out_%s_grid.mat',nest_str) ;
fullPath = [cdDir readFile];
load (fullPath) ;

disp('Done - cgemAnalyze retrieve SWAN results')
```

#### 5. interpolate swan results along decimated 5m isobath

```
[H_05_i_ufx]=interp2(Xp,Yp,Hsig,x_05_i,y_05_i);
[Dir_05_i_ufx]=interp2(Xp,Yp,Dir,x_05_i,y_05_i);

x = Dist_05_i(find(isnan(H_05_i_ufx)==0));
xi = Dist_05_i;
Y = H_05_i_ufx(find(isnan(H_05_i_ufx)==0));
H_05_i = interp1(x,Y,xi);

%H_05_i = interp1(Dist_05_i(find(isnan(H_05_i_ufx)==0)),H_05_i_ufx(...
                  find(isnan(H_05_i_ufx)==0)),Dist_05_i);
Dir_05_i = interp1(Dist_05_i(find(isnan(Dir_05_i_ufx)==0)),Dir_05_i_ufx(...
                  find(isnan(Dir_05_i_ufx)==0)),Dist_05_i);

[Depth_05_i]=interp2(Xp,Yp,Depth,x_05_i,y_05_i);
[Hsm]=cgemSmoother(sm_win_sz,H_05_i); Hsm=Hsm';
[Dirsm]=cgemSmoother(sm_win_sz,Dir_05_i); Dirsm=Dirsm';

disp('Done - cgemAnalyze interp SWAN results')
```

#### 6. calculate angle of incidence along 5m isobath

```
%[AI_05_i]=cgemComputeAngleInc(Dir_05_i,CN_05_i);
AI_05_i=Dir_05_i-CN_05_i;

%[Aism]=cgemComputeAngleInc(Dirsm,CNsm);
Aism=Dirsm-CNsm;

disp('Done - cgemAnalyze AngleOfIncidence')
```

#### 7. compute WvEnrgy, Wv.EnrgyFlx, Sxy, and Strss-FlxFactor

```
[E_05_i,P_05_i,RS_05_i,SF_05_i]=...
```

*last modified March 9, 2012*

```

    cgemComputeWvEnergyFlx(x_05_i,y_05_i,H_05_i,AI_05_i);
    [Esm,Psm,RSsm,SFsm]=cgemComputeWvEnergyFlx(x_05_i,y_05_i,Hsm,Aism);
    disp('Done - cgemComputeWvEnergyFlx')

```

## 8. compute divergence of drift along 5m isobath

```

    [IR_05_i,QR_05_i,DD_05_i]=cgemComputeDivDrift(SF_05_i,dec_spc,K);
    [IRsm,QRsm,DDsm]=cgemComputeDivDrift(SFsm,dec_spc,K);
    [DDsmasm]=cgemSmoother(coast_sm_win_sz2,DDsm);

    disp('Done - cgemComputeDivDrift')

    H_along(:,i)=H_05_i; Hsm_along(:,i)=Hsm;
    Dir_along(:,i)=Dir_05_i; Dirsm_along(:,i)=Dirsm;
    AI_along(:,i)=AI_05_i; Aism_along(:,i)=Aism;
    E_along(:,i)=E_05_i; Esm_along(:,i)=Esm;
    P_along(:,i)=P_05_i; Psm_along(:,i)=Psm;
    RS_along(:,i)=RS_05_i; RSsm_along(:,i)=RSsm;
    SF_along(:,i)=SF_05_i; SFsm_along(:,i)=SFsm;
    IR_along(:,i)=IR_05_i; IRsm_along(:,i)=IRsm;
    QR_along(:,i)=QR_05_i; QRsm_along(:,i)=QRsm;
    DD_along(:,i)=DD_05_i; DDsm_along(:,i)=DDsm; DDsmasm_along(:,i)=DDsmasm;

    save([results_dir prj_s nest_str 'output.mat'],...
        'x_05_i','y_05_i','Dist_05_i','LP_05_i','CN_05_i','CNsm',...
        'H_along','Hsm_along','Dir_along','Dirsm_along','AI_along','Aism_along',...
        'E_along','Esm_along',...
        'P_along','Psm_along','RS_along','RSsm_along','SF_along','SFsm_along',...
        'IR_along','IRsm_along','QR_along','QRsm_along','DD_along','DDsm_along',...
        'DDsmasm_along','Xp','Yp','Depth','Hsig','Dir');

    display('Finished running CGEM codes');

```

## C.9 cgemDecimateIsobath.m

**function to convert irregularly spaced isobath model output to regularly spaced.**

created by pna at uf, mar. 19, 2007

```

% modified and reduced to handle just the Isobath decimation, without
% dealing with the wave heights and directions
% by pna at UF, May 26, 2008

```

```

function [Dist_05_i,x_05_i,y_05_i]=cgemDecimateIsobath(Dist_05,x_05,y_05,dec_spc);

warning off

```

*last modified March 9, 2012*

```

ind=((find(diff(Dist_05)~=0)));
Dist_05=Dist_05(ind);
x_05=x_05(ind);
y_05=y_05(ind);

Dist_05_i=[min(Dist_05):dec_spc:max(Dist_05)];

x_05_i=interp1(Dist_05,x_05,Dist_05_i);
y_05_i=interp1(Dist_05,y_05,Dist_05_i);

warning on

```

## C.10 cgemComputeCsttrnd.m

### Contents

- FILE: cgemComputeCsttrnd.m
- 1. Calculate the alongshore "slope" (variable -> m)
- 2. Calculate alongshore uncorrected coastal trend
- 3. Calculate CORRECTED coastal trend and coast nrml.

function to determine trend (orientation) of the coastline (actually, the 5m depth contour, for use in the calculation of the stress-flux factor (i.e. the longshore component of wave power.)

```

function [CT_c,CT_u,Q,CN,m_h,m_v,range_x,range_y]=...
    cgemComputeCsttrnd(x_05_i,y_05_i,winsize,prj_s)

% where (x_05_i,y_05_i) are the interpolated, evenly-spaced positions along
% the 5m bathymetric contour (in meters w.r.t. 32N Lat, 121W Long)

% history:

% began by [pna at uf, March 2007]

% overhauled to include quadrant-based correction and fitting line w.r.t.
% latitude, as opposed to w.r.t. longitude by [pna at uf, June 2007]

% successfully fixed - now the code uses the longer of two ranges (x vs. y)
% in region to "fit" the proper trendline [by pna, in-flight, SF ->
% Charlotte, June 12, 2007]

% altered again to separate the 'computation' from the 'plotting' of
% coastal trends. now this code does the computation, whereas
% 'cgemPlotCsttrnds.m' does the plotting [by pna at uf, may 27, 2008]

warning off

```

*last modified March 9, 2012*

## 1. Calculate the alongshore "slope" (variable -> m)

```

hfwinsize=floor(winsize/2);

CT_u=zeros(1,length(x_05_i));
CT_u(1:hfwinsize)=0./0;    % Set 1st few vals of trend to NaN
CT_u(end-hfwinsize+1:end)=0./0;    % Set last few trend vals to NaN
CT_u_h=zeros(1,length(x_05_i));
CT_u_h(1:hfwinsize)=0./0;    % Set 1st few vals of trend to NaN
CT_u_h(end-hfwinsize+1:end)=0./0;    % Set last few trend vals to NaN
CT_u_v=zeros(1,length(x_05_i));
CT_u_v(1:hfwinsize)=0./0;    % Set 1st few vals of trend to NaN
CT_u_v(end-hfwinsize+1:end)=0./0;    % Set last few trend vals to NaN

for j=hfwinsize+1:length(x_05_i)-hfwinsize;
    % Define "region" (pts. up- & downcoast of current pt.) using hfwinsz
    region=[j-hfwinsize:j+hfwinsize];
    % Fit the data with a trendline, using x as indep. variable (y as dep.)
    s_i_h=polyfit(x_05_i(region),y_05_i(region),1);
    m_h(j)=s_i_h(1);
    b_h(j)=s_i_h(2);
    % Fit the data with a trendline, using y as indep. variable (x as dep.)
    s_i_v=polyfit(y_05_i(region),x_05_i(region),1);
    m_v(j)=s_i_v(1);
    b_v(j)=s_i_v(2);
    % Determine Range of data in region w.r.t. x and y
    range_x(j)=abs(max(x_05_i(region))-min(x_05_i(region)));
    range_y(j)=abs(max(y_05_i(region))-min(y_05_i(region)));
    % Define the trend line to the region
    if range_y(j)>range_x(j)
        m(j)=1./m_v(j);
        xfit(1)=m_v(j)*max(y_05_i(region))+b_v(j);
        xfit(2)=m_v(j)*min(y_05_i(region))+b_v(j);
        yfit(1)=(xfit(1)-b_v(j))/m_v(j);
        yfit(2)=(xfit(2)-b_v(j))/m_v(j);
    else
        m(j)=m_h(j);
        yfit(1)=m_h(j)*max(x_05_i(region))+b_h(j);
        yfit(2)=m_h(j)*min(x_05_i(region))+b_h(j);
        xfit(1)=(yfit(1)-b_h(j))/m_h(j);
        xfit(2)=(yfit(2)-b_h(j))/m_h(j);
    end
    % Calculate distance from first point in region to endpoints on fit
    dist_2_fitpt1=sqrt(((x_05_i(region(1))-xfit(1))^2+((y_05_i(region(1))-yfit(1))^2));
    dist_2_fitpt2=sqrt(((x_05_i(region(1))-xfit(2))^2+((y_05_i(region(1))-yfit(2))^2));
    % Compare distances from 1st pt. in region to endpoints on fit
    compare=dist_2_fitpt2-dist_2_fitpt1;
    % Reorient fit line so closer of two endpoints is "first" pt on fit
    if compare<0

```

*last modified March 9, 2012*

```

        xfit=fliplr(xfit);
        yfit=fliplr(yfit);
    end
    % Determine which direc. last fit point is w.r.t. first fit point, and
    % assign quadrant accordingly
    if xfit(1)<=xfit(2)&yfit(1)<yfit(2)
        Q(j)=1;
    elseif xfit(1)<xfit(2)&yfit(1)>=yfit(2)
        Q(j)=4;
    elseif xfit(1)>xfit(2)&yfit(1)<=yfit(2)
        Q(j)=2;
    elseif xfit(1)>=xfit(2)&yfit(1)>yfit(2)
        Q(j)=3;
    else
        disp(['j=' j ' : No Quadrant Calculated'])
    end
    clear s_i
end

range_x(1:hfwinsize)=zeros(1,hfwinsize)./0;
range_x(length(range_x)+1:length(range_x)+hfwinsize)=zeros(1,hfwinsize)./0;
range_y(1:hfwinsize)=zeros(1,hfwinsize)./0;
range_y(length(range_y)+1:length(range_y)+hfwinsize)=zeros(1,hfwinsize)./0;
m(1:hfwinsize)=zeros(1,hfwinsize)./0;
m(length(m)+1:length(m)+hfwinsize)=zeros(1,hfwinsize)./0;
m_h(1:hfwinsize)=zeros(1,hfwinsize)./0;
m_h(length(m_h)+1:length(m_h)+hfwinsize)=zeros(1,hfwinsize)./0;
m_v(1:hfwinsize)=zeros(1,hfwinsize)./0;
m_v(length(m_v)+1:length(m_v)+hfwinsize)=zeros(1,hfwinsize)./0;
b_h(1:hfwinsize)=zeros(1,hfwinsize)./0;
b_h(length(b_h)+1:length(b_h)+hfwinsize)=zeros(1,hfwinsize)./0;
b_v(1:hfwinsize)=zeros(1,hfwinsize)./0;
b_v(length(b_v)+1:length(b_v)+hfwinsize)=zeros(1,hfwinsize)./0;
Q(1:hfwinsize)=zeros(1,hfwinsize)./0;
Q(length(Q)+1:length(Q)+hfwinsize)=zeros(1,hfwinsize)./0;

```

## 2. Calculate alongshore uncorrected coastal trend

```

CT_u=(180/pi)*(atan(m));
CT_u_v=(180/pi)*(atan(1./m_v));

```

## 3. Calculate CORRECTED coastal trend and coast nrml.

```

CT_c=CT_u;
CT_c(find(Q==4))=(-CT_c(find(Q==4)))+90;
CT_c(find(Q==3))=(90-CT_c(find(Q==3)))+180;
CT_c(find(Q==2))=(-CT_c(find(Q==2)))+270;

```

*last modified March 9, 2012*



```
CT_c(find(Q==1))=(90-CT_c(find(Q==1)));  
CN=90+CT_c;  
if strcmp(prj_s,'nfa'),CN=CN-180; end  
warning on
```

## C.11 cgemComputeWvEnergyFlx.m

### Contents

- Compute wave energy
- Compute wave celerity
- Compute energy flux
- Compute radiation stress
- Compute longshore component of wave energy flux

function to compute longshore component of wave energy flux along 5m isobath.

```
function [E_05_i,P_05_i,RS_05_i,SF_05_i]=...  
    cgemComputeWvEnergyFlx(x_05_i,y_05_i,H_05_i,AI_05_i)  
  
% history:  
  
% began by [pna at uf, March 2007]  
% documented by pna at uf July 10, 2007  
% modified to remove all the plotting, by pna at uf, may 29, 2008  
  
warning off  
  
rho=1024;  
g=9.81;  
depth=5;  
n=1;
```

### Compute wave energy

```
E_05_i=(1/8)*rho*g*(H_05_i.^2);
```

### Compute wave celerity

```
C_05_i=ones(1,length(E_05_i)).*sqrt(g*depth);
```

### Compute energy flux

```
n_05_i=ones(1,length(E_05_i)).*n;  
P_05_i=E_05_i.*C_05_i.*n_05_i;
```

### Compute radiation stress

```
RS_05_i=E_05_i.*n_05_i.*cosd(AI_05_i).*sind(AI_05_i);
```

*last modified March 9, 2012*

## Compute longshroe component of wave energy flux

```
SF_05_i=P_05_i.*cosd(AI_05_i).*sind(AI_05_i);
```

```
warning on
```

## C.12 cgemComputeDivDrift.m

function to compute divergence of longshore drift along 5m isobath.

### Contents

- Compute Transport Rate
- Computer Volumetric Transport Rate
- Compute Divergence of Drift

```
function [IR_05_i,QR_05_i,DD_05_i]=cgemComputeDivDrift(SF_05_i,dec_spc,K)
```

```
% history:
```

```
% began by [pna at uf, March 2007]
```

```
% documented by pna at CEC conf., Sept., 2007
```

```
% modified/separated from plotting portion by pna at UF, May 29, 2007
```

```
warning('off','MATLAB:divideByZero')
```

```
rho_sed=2650;
```

```
rho_wat=1024;
```

```
g=9.81;
```

```
No=0.6;
```

### Compute Transport Rate

```
IR_05_i=K.*SF_05_i;
```

### Computer Volumetric Transport Rate

```
QR_05_i=IR_05_i./((rho_sed-rho_wat).*g.*No); % in m3/sec
```

### Compute Divergence of Drift

```
DD_05_i=[0 -diff(QR_05_i)]/dec_spc;
```

```
warning on
```

*last modified March 9, 2012*

# Index

*AI\_05\_i*, 23, 27  
*alp*, 12, 15, 27  
*alpn*, 17  
  
*base\_dir*, 12, 27  
*batch\_num*, 14  
*bathy\_grd\_name*, 12, 15, 17, 27  
*bathy\_grd\_params*, 17  
*bathy\_grd\_params\_s*, 27  
  
*cgemAnalyze.m*, 19, 26, C-12  
*cgemComputeCsttrnd.m*, 22, C-16  
*cgemComputeDivDrift.m*, 24, C-20  
*cgemComputeWvEnergyFlx.m*, 23, C-19  
  
*cgemDecimateIsobath.m*, 21, C-15  
*cgridscale*, 15  
*CN*, 22, 27  
*coast\_sm\_win\_sz1*, 19  
*coast\_sm\_win\_sz2*, 19  
*CT\_c*, 22, 27  
*CT\_u*, 22, 27  
  
*DD\_05\_i*, 24, 27  
*dec\_spc*, 19, 21, 24, 27  
*Dfix*, 27  
*Dir*, 14, 15, 17, 19, 27, A-4  
*Dir\_all*, 14  
*Dir\_range*, 12, 27  
*Dist\_05*, 21, 27  
*Dist\_05\_i*, 21  
*Dist\_i*, 28  
*dxinp*, 13, 15, 26, 28  
*dyinp*, 13, 15, 26, 28  
  
*E\_05\_i*, 23, 28  
  
*first\_runnum*, 14  
  
*gridspacing*, 13, 26, 28  
  
*H*, 19, 28  
*H\_05\_i*, 23, 28  
*Hfix*, 28  
*Hs*, 14, 15, 17, 28, A-4  
*Hs\_all*, 14  
*Hs\_range*, 12, 28  
  
*i\_vals*, 16, 28  
*input\_dir*, 12–14, 19, 28  
*IR\_05\_i*, 24, 28  
*isobath\_dir*, 19  
  
*j\_vals*, 16, 29  
  
*K*, 19, 24, 29  
  
*m\_h*, 22, 29  
*m\_v*, 22, 29  
*method*, 29  
*mx*, 12, 15, 29  
*mx\_c*, 16, 29  
*mxn*, 17  
*my*, 12, 15, 29  
*myc*, 16, 29  
*myn*, 17  
  
*nest\_id*, 15, 17, 18, 29  
*nest\_names*, 12, 15, 29  
*nest\_str*, 17, 19  
*nest2run*, 12, 29  
*new\_dir*, 15, 17, 29  
*num\_cols*, 13, 15, 26, 30  
*num\_processes*, 12, 18, 30  
*num\_rows*, 13, 15, 26, 30  
  
*output\_dir*, 19  
  
*P\_05\_i*, 23, 30  
*prj\_s*, 12, 13, 15–19, 30  
*projectName*, 12, 15, 17, 30

*Q*, [22](#), [30](#)  
*QR\_05\_i*, [24](#), [30](#)  
  
*range\_x*, [22](#), [30](#)  
*range\_y*, [22](#), [30](#)  
*RS\_05\_i*, [23](#), [30](#)  
*run\_num\_s*, [15](#), [17](#), [18](#), [30](#)  
*run\_nums*, [14](#)  
  
*SF\_05\_i*, [23](#), [24](#), [30](#)  
*sm\_win\_sz*, [19](#)  
*spread*, [12](#), [15](#), [17](#), [30](#), [A-4](#)  
*swanCalcIJVals.m*, [16](#)  
*swanControl.m*, [12](#), [C-4](#)  
*swanDwwvmaker.m*, [14](#), [C-8](#)  
*swanExecuter.m*, [18](#)  
*swanInputWriter.m*, [15](#), [C-9](#)  
*swanNestWriter.m*, [17](#)  
*swanPrjDetails.m*, [13](#), [26](#), [C-7](#)  
  
*T*, [14](#), [15](#), [17](#), [19](#), [30](#), [A-4](#)  
*T\_all*, [14](#)  
*T\_range*, [12](#), [31](#)  
  
*user*, [31](#)  
  
*wetsedExecuter.m*, [11](#), [C-1](#)  
*wetsedInputVariables.m*, [C-2](#)  
*wetsedNestDetails.m*, [C-3](#)  
*win\_size*, [22](#), [31](#)  
  
*x\_05*, [21](#), [26](#), [31](#)  
*x\_05\_i*, [21–23](#), [31](#)  
*xlen*, [12](#), [15](#), [31](#)  
*xlenn*, [17](#)  
*xp*, [12](#), [15](#), [31](#)  
*xpn*, [17](#)  
  
*y\_05*, [21](#), [26](#), [31](#)  
*y\_05\_i*, [21–23](#), [31](#)  
*ylen*, [12](#), [15](#), [31](#)  
*ylenn*, [17](#)  
*yp*, [12](#), [15](#)  
  
*ypn*, [17](#)  
  
*Z*, [26](#)