Topo-bathymetric lidar total propagated uncertainty (TPU) subaqueous portion Monte Carlo simulations

by

Firat Eren

firateren85@gmail.com

Center for Coastal and Ocean Mapping

University of New Hampshire

March 1, 2019

1. Introduction

The goal of this document is to describe the code written as part of the subaqueous portion of the topo-bathymetric lidar total propagated uncertainty (TPU) model. The overall goal of the subaqueous portion of the TPU model is to characterize the interaction of the laser ray during its travel from the water surface to the seafloor and understand its effect on the measurement uncertainty.

The main interactions of the laser beam in the subaqueous portion is with the water surface, i.e. refraction, and water column, i.e. scattering and absorption. However, the complexity is that there is no available analytical model that describes the radiometric and the geometric interactions of the light with the water surface and the water column. Due to lack of deterministic approaches, a stochastic approach, i.e. Monte Carlo simulation, is adopted to develop subaqueous TPU model. By analyzing the laser beam interaction with its environment under a variety of environmental conditions and with a large number of simulation runs, it is possible to make vertical and horizontal measurement uncertainty assessments with Monte Carlo approach.

The focus of the Monte Carlo simulations is the laser beam interaction with the water surface and water column. Ray-tracing methods are adopted, and the laser rays are traced until they intersect with the seafloor. Reflection from the seafloor is not taken into account. Similarly, atmospheric losses are not taken into account during the development of the model. Although the focus of the subaqueous TPU model is on the laser beam intersection with its environment, i.e. the water surface and water column, laser scanning geometry and survey configurations also have an impact on the depth measurement. Factors such as flight altitude, laser beam footprint on the water surface and the laser beam off-nadir angle are also taken into account in the developed TPU model. Presented subaqueous TPU model is developed based on the Riegl VQ-880-G system configurations. However, different lidar manufacturer designs can be easily configured in the Monte Carlo simulation code.

The programs written for the Monte Carlo simulations are introduced in this manual. These programs are also uploaded to the TPU team's Box website. Although the code is written to

include as many comments as possible, some details that might be missing in the provided codes are provided in this document. A potential questions section (Section 3) is also added to answer potential questions that might arise to run the code, data formats and post-processing of the data.

References used for Monte Carlo simulation

The main references used for the Monte Carlo simulation programs are Curtis Mobley's book Light and Water as well as Mobley's tutorial on 3D water surface generation. Paper written by DeGreve, 2016 (Reflections and Refractions in Ray Tracing) is also found to be a very useful reference for ray-tracing basics. Similarly, for three-dimensional water surface generation, the tutorial, Modeling Sea Surface: A tutorial on Fourier Transform Techniques, written by Mobley should be an excellent reference. The theoretical details of the background material needed to write the document are omitted in this document. The user/developer is strongly recommended to read or at least refer to the following references for a better understanding of the background material. The following references are also uploaded to the Box website.

Book Chapters:

- Light and Water: Radiative Transfer in Natural Waters- Chapter 3: Optical Properties of Water (Curtis D. Mobley). This chapter describes the basics of the
- Light and Water: Radiative Transfer in Natural Waters- Chapter 6: Monte Carlo simulations (Curtis D. Mobley).

Tutorials

- Modeling Sea Surfaces: A tutorial on Fourier Transform Techniques (Curtis D. Mobley). This is an excellent source on the Fourier Transform techniques regarding the generation of 3D water surface models.
- Ocean Optics Web book. This is also a good source on Ocean Optics with good explanations on the concepts such as volume scattering and Monte Carlo simulations. http://www.oceanopticsbook.info/
- DeGreve, B. 2016. Reflections and Refractions in Ray Tracing. URL: http://graphics.stanford.edu/courses/cs148-10-summer/docs/2006--degreve--reflection_refraction.pdf

Journal papers

- F. Eren, Jung, J., Parrish, C. E., Forfinski, N., and Calder, B. R., "Total Vertical Uncertainty (TVU) modeling for topo-bathymetric lidar systems", American Society for Photogrammetry and Remote Sensing (ASPRS). In Press. This journal paper is written by the TPU project team and one of the products of the project.
- "A unified directional spectrum for long and short wind-driven waves." T. Elfouhaily, B. Chapron, K. Katsaros, and D. Vandemark in J. Geophys. Res. 102:15781-15796, 1997. This journal paper introduces the ECKV wave model used in this study.
- "Correlation of beam and diffuse attenuation coefficients measured in selected ocean waters", Shannon, J. G. International Society for Optics and Photonics In Ocean Optics IV 1975, 63, 3-12. This paper reports results of the experiments conducted in a variety of water

clarity types to determine the relationship between the beam attenuation coefficient and the diffuse attenuation coefficient.

2. Monte Carlo simulation code

The Monte Carlo simulation code was written in MATLAB. A modular approach was adapted in the Monte Carlo simulation code. These modules describe the following main components:

1) Laser ray generation and its travel from the aircraft until water surface. This step is done for ray-tracing calculations for refraction only and should not be confused with the sub-aerial portion of the TPU model.
2) Three-dimensional water surface model generation. First interaction of the laser beam is with the water surface. Therefore, three-dimensional water surface model is generated in this section to calculate the refraction of the laser rays into the water column.
3) Scattering within the water column. Scattering process within the water column is modeled. The simulations continue until the laser ray intersects with the seafloor.

The following programs are written in MATLAB to model the processes mentioned above. There is a total of eight programs (script, functions and a class).

1) Lidar_TPU_MC_subaqueous.m – Main script file
2) GenerateLidarRays.m- Function
3) SurfaceSpectrum.m - Class
    a. Cos2S_spread.m - Function
    b. ECKV_spectrum.m - Function
4) SimulateShot.m- Function
5) FormatTimeString.m - Function
6) TateBryan.m- Function

The main program that runs the subaqueous portion of the Monte Carlo simulation is **Lidar_TPU_MC_subaqueous.m**. This program is the only script file and calls the other seven functions/classes as illustrated in Figure 1. The details of these programs are provided in the following sections.

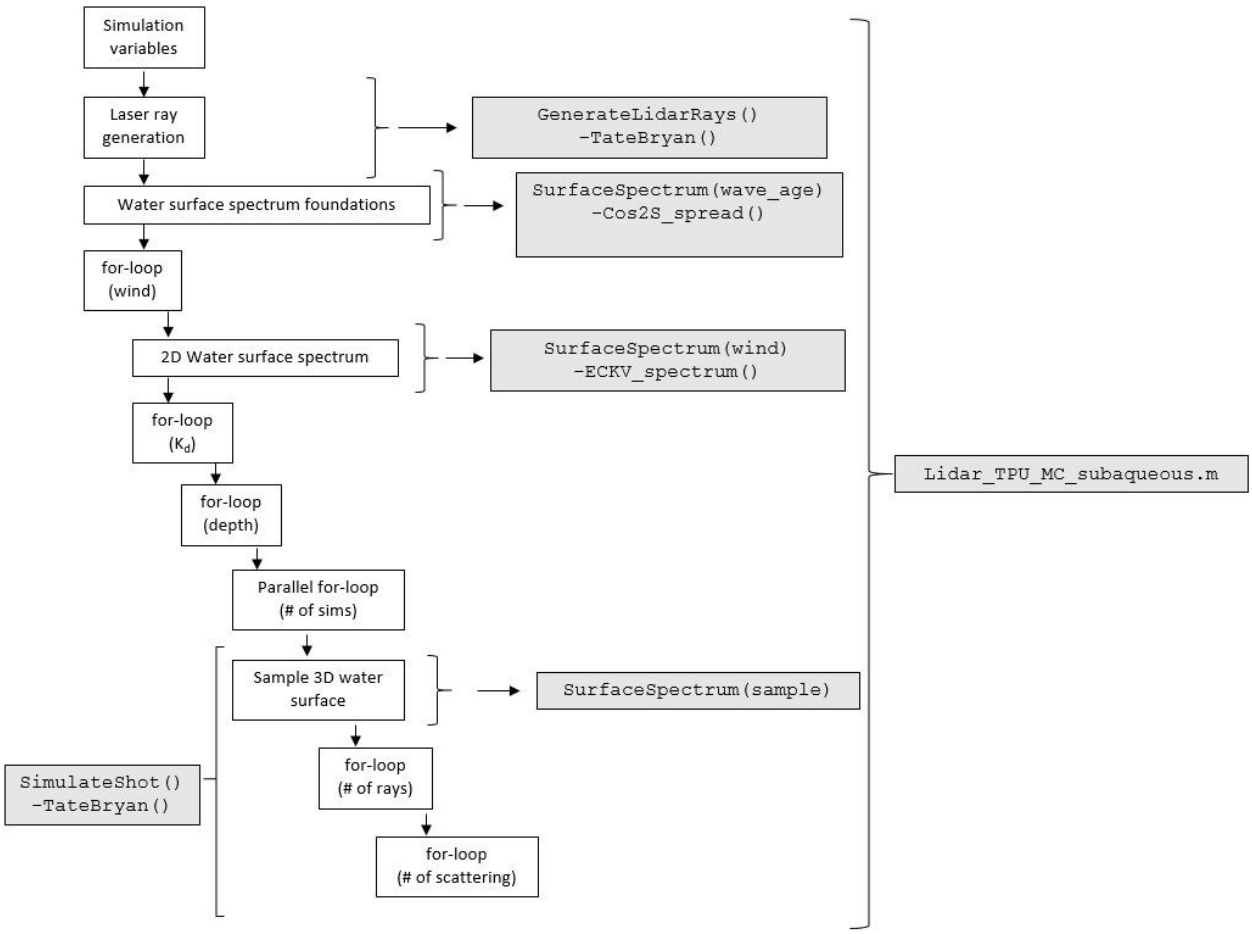

Figure 1. Monte Carlo simulation schematic

2.1. Lidar_TPU_MC_subaqueous.m

This is the core program of the Monte Carlo TPU calculations and the program to run to obtain TPU results. The user needs to click the run button on the program to start the simulations. After the simulations are completed, the results will be automatically saved into the MATLAB workspace as a MATLAB file.

The program consists of six for-loops (**for** each wind speed, **for** each Kd value, **for** each depth, **for** the number of runs, **fo**r the number of rays and **for** each scattering event).In the beginning of the program user defined parameters as well as their function within the simulation and the current values used in the Monte Carlo simulation program are introduced.

2.1.1. Simulation properties (`AlgConst.Sim`)

In this section, main simulation parameters are defined. The default values are also shown.

`AlgConst` is the structure-of-structures variable to define the parameters to be used in the MC simulations. For example, **`AlgConst.Sim`** stores all the simulation related parameters including:

1) Number of laser rays with the laser beam (**AlgConst.Sim.Nrays**): This is a unitless parameter. Value used in the current set of results is 1000.
2) Maximum scatter events (**AlgConst.Sim.MaxScatterEvents**). Describes the number of scattering layers within the water column and it is a unitless parameter. In underwater, laser rays undergo scattering. Value used in the current set of results is 20.
3) Number of simulations to run (**AlgConst.Sim.Nsim**): How many simulations is it going to be run to generate results? Value used in the current set of results is 2000.
4) Wind speed (**AlgConst.Sim.wind_spread**): Wind speed range to be used in simulations (units: $\frac{meters}{seconds}$). In the simulations, this variable is set to 1 to 10 m/s with 1 m/s intervals. This variable is an input to the water surface generation program (**SurfaceSpectrum.m**)
5) Diffuse attenuation coefficient (**AlgConst.Sim.Kd_spread**): Diffuse attenuation coefficient is an indication of water clarity. This value is used to calculate beam attenuation coefficient in the simulations as the laser ray scattering calculations use the beam attenuation coefficient rather than diffuse attenuation coefficient. The higher the diffuse attenuation coefficient, the smaller the path between each scattering event resulting to an increased scattering. Thus, higher laser beam spread is expected with higher diffuse attenuation coefficient values.
6) Depth range (**AlgConst.Sim.drange)** . Depth range used in the simulations. Set at 0.1 m increments. Higher increments could be used for speed improvements.

*2.1.2. Scattering parameters within the water column (**AlgConst.Scatter**)*

1) Single scattering albedo (**AlgConst.Scatter.wo**): Single scattering albedo is an inherent optical property (IOP) and defines how much of the light absorption is due to scattering and how much of it is due to absorption. This is a unitless parameter and vary between 0-1. Higher values denote scattering dominated water column. The current value set for **AlgConst.Scatter.wo** is 0.80.
2) Henyey-Greenstein phase function forward scattering parameters (**AlgConst.Scatter.g_pf**): This parameter controls if the medium is forward or backward scattering dominated. Numbers close to 1 denote more forward scattering dominated water column. This parameter is set to 0.995. (http://www.oceanopticsbook.info/view/scattering/the_henyeygreenstein_phase_function).
3) Henyey-Greenstein scattering phase function constant 1 (**AlgConst.Scatter.hg_const_1**). Constant used to calculate the scattering
4) Henyey-Greenstein scattering phase function constant 2 (**AlgConst.Scatter.hg_const_2**).

2.1.3. *Environmental parameters (**AlgConst.Env**)*

1) Index of refraction in air (**AlgConst.Env.air_refraction_index**): This value is set to 1.00. Unitless parameter.
2) Index of refraction in water (**AlgConst.Env.water_refraction_index**). This value is set to 1.33. Unitless parameter.
3) Wave age (**AlgConst.Env.wave_age**). Wave age is a parameter that is used to determine the age of the surface waves in the ECKV model. This is a unitless parameter. This value can be picked up based on the sea-state as follows:

   = 0.84 for a fully developed sea (corresponds to Pierson-Moskowitz wave model)
   = 1 for a "mature" sea
   = 2 to 5 for a "young" sea; the maximum allowed value is 5.
4) Shallowest depth (**AlgConst.Env.dshallow**). This is the shallowest depth in meters. This value is set to -1 m where 0 m is the mean water surface elevation.
5) Deepest depth (**AlgConst.Env.ddeep**). This is the deepest depth in meters. This value is set to -10 m where 0 m is the mean water surface elevation.
6) Water surface elevation (**AlgConst.Env.water_elevation**). This is set to 0 m.

2.1.4. *Lidar properties (**AlgConst.Lidar**)*

1) Laser beam off nadir angle (**AlgConst.Lidar.scan_angle**). Off nadir scan angle for Riegl Vq-880-G system in degrees. This is set to 20 degrees based on Riegl VQ-880-G spec sheet.
2) Laser beam divergence angle (**AlgConst.Lidar.beam_div**). Half beam divergence angle in milliradians (mrad). This value is between 0.7-2.0 mrad as defined in Riegl VQ-880-G spec sheet. This is converted to degrees later in the program.
3) Distance vector between the IMU and the laser unit (**AlgConst.Lidar.PG**). This is the distance vector from the aircraft IMU to the laser unit. This is the level arm offset vector. The unit is in meters. In the current version, this vector is set to [0;0;0]. That is, it is assumed that the IMU and the laser are in the same location.
4) Sensor boresight angle, $\Delta\kappa$ (**AlgConst.Lidar.dkap**). Angle in degrees. Set to 0 degrees in the simulation.
5) Sensor boresight angle, $\Delta\psi$ (**AlgConst.Lidar.dpsi**). Angle in degrees. Set to 0 degrees in the simulation.
6) Sensor boresight angle, $\Delta\omega$ (**AlgConst.Lidar.domeg**). Angle in degrees. Set to 0 degrees in the simulation.

2.1.5. *Position and orientation of the aircraft*

1) Laser position (**AlgConst.PosAtt.laser_location**) This is the vector that determines the aircraft position in Cartesian coordinates (x, y, z). Units are in meters. This vector is set to [0, 0, 600] denoting that the aircraft elevation is 600 m above the water surface elevation and 0 m in x and y-axes.
2) Aircraft roll (**AlgConst.PosAtt.roll**). Roll of the aircraft in degrees. Set to 0 degrees in the simulation.

3) Aircraft pitch (**`AlgConst.PosAtt.pitch`**). Pitch of the aircraft in degrees. Set to 0 degrees in the simulation.
4) Aircraft yaw (**`AlgConst.PosAtt.yaw`**). Yaw of the aircraft in degrees. Set to 0 degrees in the simulation.

After the simulation variables are set, next step is to generate the Laser Rays that are emitted from the aircraft and incident on the water surface. The purpose of this step is to calculate the refraction angles of the laser rays with the water surface. The name of the MATLAB script file to generate the lidar rays is **GenerateLidarRays.m**

2.2. GenerateLidarRays.m

GenerateLidarRays.m is a function that takes algorithm constants (**`AlgConst`**) as an input. The output of the function is the position of the laser (`pos`) and the laser ray locations in x and y axis (**`ray_x_pos`** and **`ray_y_pos`**). In this function, another function, namely **TateBryan.m**, is called to calculate the rotation matrix to be used in the laser position calculation. Then, the laser beam diameter is calculated (**`diam`**) based on the laser unit location and the laser beam off-nadir and divergence angle. After the laser beam footprint diameter is determined, laser beam rays whose coordinates are randomly assigned are created.

Important! Another option is to generate a set of laser ray data once and use it many times for the rest of the simulations [1]. This makes it possible to use the same set of laser rays for different environmental conditions.

2.3. SurfaceSpectrum.m

Next step of the TPU program is to generate the 3D water surface. Here, **SurfaceSpectrum.m** class is created. Creating a 3D water surface model can be computationally intensive in Monte Carlo simulations as there can be a significant number of runs, more than a million. In order to save computation time, three functions are called in different places in the program.

1) SurfaceSpectrum (wave_age)
2) SetWind (obj, wind)
3) Sample (obj, sample)

2.3.1. SurfaceSpectrum (wave_age)

The input to this function is the wave age and the output is a surface spectrum object that is later passed to the other functions in the program. The following objects are calculated in this function, wave age (passed from the main simulation program), kfx (fundamental frequencies in x-axis), kfy (fundamental frequencies in y-axis), k (spatial variable needed to generate the

[1] Instead of randomly assigning laser beam footprints anytime **GenerateLidarRays.m** function is called, another option is to pre-compute the laser ray locations and use it many times. By doing this, the laser ray location is held constant and the effect of changing environmental parameters on the TPU calculations are better observed. If this option is chosen, the user should make sure to deactivate (comment out) the **GenerateLidarRays.m** code and load the laser ray location file. It is denoted as rays_1000 at the beginning of the **Lidar_TPU_MC_subaqueous.m** program. Also, the **`POS_LAS`** variable should be defined. This line is below the **GenerateLidarRays.m** function.

modeled wave spectrum, i.e. ECKV wave spectrum[2]), SF (spreading function generated in Cos2S_spread function), scaled_frequency_bins and domain_triangulation (used in triangulation of the 3D water surface elevation model). The surface wave spectrum depends only on the wave age and wind for the most part. Therefore, to avoid re-computation, this function is only called once within the main program (**Lidar_TPU_MC_subaqueous.m**) outside of the inner for-loops. We take into account the wave age now and the whole wave spectrum is generated when wind speed is taken into account in **SetWind(obj, wind)** function as explained below.

2.3.2. SetWind (obj, wind)

The input to this function is the surface spectrum object and the wind speed. In this function, ECKV wave model spectrum is called to generate the 1-D omnidirectional wave spectrum object. Then, this is converted to 2D wave spectrum which is the final step needed to generate 3D water surface elevation model. This is also the output of the program and the input needed to generate a 3D water surface elevation model. Because wind speed determines the wave spectrum, this function is called within the wind speed loop in the main program (**Lidar_TPU_MC_subaqueous.m**). A new 2D wave spectrum is generated for each wind speed.

2.3.3. Sample (obj, t)

The input to this function is surface spectrum object and sample at t=0. This function generates a different 3D water surface elevation model each time it is called. Therefore, it is called within the innermost loop in the **SimulateShot.m** function. FFT methods are used to generate 3D water surface model in this function. After water surface elevation model is generated, Delaunay triangulation methods are applied to partition the water surface model in triangles. Then, the face normal of these triangles are calculated to calculate the refraction angle of the laser rays into the water column. The background material for this section can be found in the following tutorial: Modeling Sea Surfaces: A tutorial on Fourier Transform Techniques (Curtis D. Mobley).

2.4. SimulateShot.m (depth, cb, POS_LAS, x_rays, y_rays, surface_spectrum, AlgConst)

This function traces the laser ray through the water column until it intersects with the seafloor. It should also be noted that this function is run in parallel for-loop (parfor) as this is the most time-consuming part of the program. The output of this function is a 3x1 vector that gives the mean positions of each laser ray, i.e. [x, y, z]. The program starts with the calculation of the scattered ray lengths based on the beam attenuation coefficient value, which is calculated from the diffuse attenuation coefficient[3]. Then, a random number is generated. If this random number is lower than the single scattering albedo determined at the beginning of the program (0.80), then the event is a scattering event. Otherwise, the event is an absorption and the corresponding ray is terminated. Scattering angles in azimuth and elevation are calculated. The refraction calculations are conducted by using ray tracing equations[4] and by using the face normal values generated

[2] The acronym ECKV is the first names of the authors of the paper: "A unified directional spectrum for long and short wind-driven waves." T. Elfouhaily, B. Chapron, K. Katsaros, and D. Vandemark in J. Geophys. Res. 102:15781-15796, 1997.

[3] See Shannon 1975 paper for the relationship between diffuse attenuation coefficient and the beam attenuation coefficient.

[4] DeGreve paper is a good source for refraction and reflection calculations.

from Sample(obj, t) function. Here, we basically calculate the incidence angle of each ray with the corresponding water surface facet. After this step, the ray enters the water column for scattering calculations.

Water column is the section where two for-loops is used. The outer for-loop is for each laser beam ray. The inner for-loop for each scattering event. Here, scattering calculations are conducted by using the scattered ray lengths. The scattering angles calculated outside of the loops are used in rotation matrices, in TateBryan() functions to calculate the new ray location. If the new ray location is higher than the set depth value in the simulation configurations, then the process is finished and path for the next ray is calculated using the same steps described above.

2.5. FormatTimeString.m (remaining_time)

This function calculates the computation time and formats the time based on the amount of time it takes. For example, if the simulation time exceeds seconds but less than a minute, the user will be informed the time in seconds. If the time is within hours but less than a day, then the computer will show the time in hours, etc. The main purpose of this function is to inform the user regarding the elapsed and remaining time for the simulations to complete. Because this function is not a core component of the Monte Carlo simulation, it is not shown in Figure 1.

3. Potential questions

a. How does the program run?

The program runs by clicking on run button in the **Lidar_TPU_MC_subaqueous.m.** Before running the program, make sure that simulation parameters are entered correctly. Process can be cancelled by pressing Ctrl+C in MATLAB workspace any time during the simulation.

b. What is the input to the Monte Carlo simulation program?

The input to the program is user defined variables and constants that are in the beginning of the **Lidar_TPU_MC_subaqueous.m** program.

In the default settings, parameters are configured to be:

Wind speed range: 1 to 10 m/s at 1 m/s increments

Diffuse attenuation coefficient range, K_d[5]: 0.06:0.01:0.36 m^{-1}.

Water depth range: 1-10 m at 0.1 m increments[6].

c. What is the output of the program? What is the data format?

The output of the program is a MATLAB data file. The output is in the form of a look-up-table (LUT) as shown in Table 1.

[5] This range is based on (Shannon, 1975) paper.

[6] Although 0.1 m increment is used, values larger than 0.1 m could be used for computational purposes.

Table 1. Sample output look-up table.

Depth (m)	Mean x (m)	Std x (m)	Mean y (m)	Std y (m)	Mean z (m)	Std z (m)
-10.00	219.75	0.55	0.00	0.64	-10.63	0.20
-9.90	219.74	0.55	0.00	0.63	-10.51	0.19
-9.80	219.72	0.55	0.00	0.62	-10.41	0.20
-9.70	219.71	0.54	0.00	0.62	-10.29	0.19
-9.60	219.69	0.55	0.00	0.61	-10.18	0.20
-9.50	219.68	0.54	0.00	0.60	-10.07	0.19
-9.40	219.67	0.52	0.00	0.61	-9.97	0.19
-9.30	219.65	0.53	0.00	0.59	-9.85	0.18
-9.20	219.64	0.53	0.00	0.58	-9.75	0.19
-9.10	219.63	0.51	0.00	0.58	-9.64	0.18

The main output of the simulation is the standard deviations in x, y and z-axes (highlighted in yellow in Table 1). It is important to note that the standard deviation values are provided in 1σ. The naming convention of the data is set to be:

table_wind_6_Kd30_ PF0.995_wave_age2

This format shows the configuration of the data file. For example, in the file name above, the simulations are for wind speed of 6 m/s, K_d value of 0.30 m^{-1}, forward scattering phase function (PF) of 0.995 and wave age 2. Notice that in the look-up table includes standard deviation values for each depth. So, for a full simulation configuration, for wind speed of 1 to 10 m/s at 1 m/s increment and K_d value of 0.06 to 0.36 at 0.1 m^{-1} increments, the number of data files will be a total of 310 files. Each data set will be named for the changing wind speed and K_d with wave age and forward scattering phase function stays the same.

d. How is the data post-processed? How can I make the data useful?

After the simulations are run and the data is saved, the next step is to analyze the data. To do this, a MATLAB script file was written, named **look_up_table_gen.m**. This file loads the corresponding MATLAB data file and implements 2nd order polynomial fit to the data (Figure 2). The user is also given the option to plot the data by switching the variable `plots='on'`, or `plots='off'`.

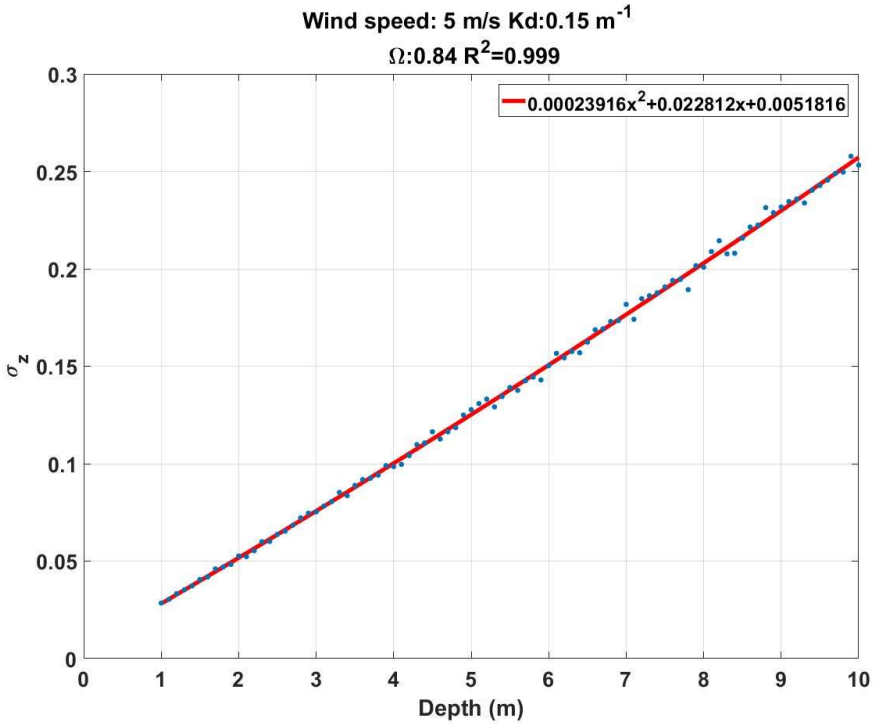

Figure 2. 2nd order polynomial line applied to the σ_z values as a function of depth. Blue dots denote the data points, red line denotes the regression line.

The result of the regression is 2nd order polynomial constants are saved as a comma separated variables (.CSV) file which is later used as an input in the cBLUE software written in Python, specifically `subaqueous.py`. Based on the wind speed and the K_d values, the program averages the polynomial coefficients

4. Speed enhancements

Monte Carlo simulations take a significant amount of time, almost about a week when run on parallel processing on a quad core, 3.6 GHz computer. To reduce the computation time, two main steps were taken: 1) Multiple water surface facet intersection and 2) switching to quaternions from Euler angles for scattering simulations. The enhanced codes are also uploaded to Box site.

4.1. Multiple water surface facet intersection

In the Monte Carlo simulation demonstrated, one water surface normal was used to calculate the laser ray refraction path. It is possible to adapt laser beam intersection with multiple water surface facets to create a more realistic refraction simulation. As the wind speed changes, the slope of the water surface facets changes, too. However, when we use the same laser ray geometry on the water surface[7], the laser rays always intersect with the same water surface ID number. Therefore, in the Monte Carlo simulation code, we determine the ID of the water surface facets that intersect with the ray outside of the main simulation loop rather than

[7] See Section 2 on GenerateLidarRays.m and footnote 1.

calculating the intersection point of the laser ray with the water surface for every time wind speed, water clarity or depth changes. This saves significant amount of computation time, considering that the Monte Carlo simulations are run approximately for a million times.

4.2. Computation of quaternions rather than Euler angles

In the subaqueous Monte Carlo simulations, rotation matrices (with TateBryan() function) were used to determine the laser ray direction vector after each scattering event within the water column (The corresponding code is in **SimulateShot.m**). The scattering ray path was multiplied with direction vector obtained through rotation matrices in yaw, pitch and roll. Although this approach is physically intuitive, i.e. it is relatively straightforward to interpret the roll, pitch and yaw angles, it brings a heavy computational burden to the simulations. In order to reduce the simulation time, quaternion approach was adopted. Preliminary results showed that the simulation time improved by 30% when compared to using rotation matrices.

5. Further recommendations

Another time-consuming part of the Monte Carlo simulations is the direction vector calculation for scattering within the water column. In its current form, direction vectors are calculated at each step. Further recommendation for enhanced computation performance is to reduce the computation time in this part of the code (**SimulateShot.m**), such as pre-computation of these coordinates and using them later, etc.