Virtual Timeseries EXperiment (VirTEX) - Quick Start

John C. Peterson
Michael B. Porter

8 July, 2011

1 Introduction

The *VirTEX* family of algorithms (short for Virtual Timeseries EXperiment) were designed to model the propagation through the underwater sound channel of a known timeseries transmitted from a hypothetical source. The algorithm computes the timeseries that would be observed at a hypothetical receiver, taking into account the effects of multipath and doppler introduced by environmental motion. The algorithms operate by post processing the outputs produced by the BELLHOP ray tracing code.

It should be noted that the *VirTEX* family of algorithms do not attempt to model every conceivable aspect of the underwater sound channel. There are many effects (e.g. volume scattering) that the models do not attempt to address.

These algorithms were designed primarily for scientists and engineers interested in the modeling of the propagation of underwater acoustic communications. (However, they could certainly be useful to researchers with other related interests).

This document is designed to assist users in getting started with the use of these algorithms. The history, capabilities and limitations of the different members of the *VirTEX* family of algorithms are briefly described in section two. Section three presents an overview of the pre-requisites and the remaining sections present outlines of running the algorithms (all written in the MATLAB interpreted language).

2 Variations of the VirTEX Algorithm

This section briefly describes the capabilities and limitations of the different members of the *VirTEX* family of algorithms. A tutorial of their use in modeling practical problems will be presented later.

2.1 Full VirTEX

The *VirTEX* algorithm was first documented in the journal paper; *Modeling broadband ocean acoustic transmissions with time-varying sea surfaces*, M. Siderius, M. Porter, Journal of the Acoustical Society

of America, vol. 124, no. 1, pp. 137-150. New variants of this algorithm have since been developed, so it is also referred to as *Full VirTEX* in this document to avoid confusion.

Full VirTEX is capable of handling almost any type of steady and unsteady environmental motion. One notable exception is the modeling of breaking waves, (this limitation is really because the BELLHOP ray tracing code assumes that the sea surface heights can be described by a single valued function of position and time, an assumption that is violated in the case of breaking waves).

The basic idea underlying *Full VirTEX* is to approximate the motion of the environment itself by a sequence of snapshots or freeze frames. A BELLHOP ray tracing computation is performed for *each* snapshot or frame, and the results are assembled to obtain a piecewise constant representation of the generalized arrival function. For practical problems with typical environmental motion, large numbers of frames are required (many thousands is not an uncommon number), making it by far the most demanding of computational resources of the *VirTEX* family of algorithms.

2.2 VirTEX for Platform Motion

VirTEX for Platform Motion was designed to address *steady motion of the source, and/or the receiver.* For each eigenray, the steady motion manifests itself as a uniform resampling (or stretching) of the known waveform transmitted by the source. The resampling can be accomplished very efficiently by pre-computing a "library" of pre-stretched waveforms using the Chirp Z transform and then performing a look-up for each eigenray. As a result, *VirTEX for Platform Motion* requires relatively modest computational resources.

2.3 VirTEX for Sea Surface Motion

VirTEX for Sea Surface Motion is capable of addressing steady motion of the source, and/or the receiver, as well as unsteady sea surface motion. The algorithm addresses the doppler introduced by unsteady sea surface motion by reading the eigenray data produced by BELLHOP. At each interaction of an eigenray with the moving sea surface, first order adjustments are made to the arrival times to account for the change in the path length from the source to receiver The computational resource requirements of *VirTEX for Sea Surface Motion* are greater than those of *VirTEX for Platform Motion*, but still significantly less than those of *Full VirTEX*.

3 Pre-Requisites

The requisite model inputs and other software packages for running *VirTEX* are briefly described in this section. A more detailed discussion of pre-requisites specific to the different versions of *VirTEX* will be presented in the tutorial sections that follow.

MATLAB software

All of the variants of the *VirTEX* algorithm were written in MATLAB, so a licensed copy of that

software is required. The OCTAVE interpreter, available at no cost under the GNU General Public License may also work (but has not been tested by the authors as of this writing).

The BELLHOP ray tracing program.
As noted above, all of the variants of the *VirTEX* algorithm operate by post-processing the output of the BELLHOP ray tracing program. While the underlying algorithm is compatible with any ray tracing software, this implementation currently supports *only* outputs produced by BELLHOP. The latest copy of BELLHOP is available free of charge under the terms of GNU General Public License from the Ocean Acoustics library hosted by HLS Research, Inc. See `http://oalib.hlsresearch.com/Rays/index.html`

Software for modeling the sea surface height.
While the *Full VirTEX* and the *VirTEX for Sea Surface Motion* algorithms were designed to model the effects of unsteady sea surface motion, they do *not* include any specific capability for modeling of the sea surface motion itself. An optional resource that the authors have found to be helpful in this regard is the WAFO (Wave Analysis for Fatigue and Oceanography) toolbox that is also written largely in MATLAB. It can be downloaded from `http://www.maths.lth.se/matstat/wafo/`. (When installing WAFO, it suffices to just unpack the files, you *do not* need to build the MEX objects). For those users that have their own software for modeling sea surface heights, the WAFO toolbox is not required.

A description of environment of interest.
The most important ingredient for successful acoustic propagation modeling is access to environmental information for the site of interest. These include, but are not limited to bathymmetry, sound speed profile data, and bottom scattering properties. These items are not needed by *VirTEX* itself, but are a pre-requisite for the preparation of the input files for running the BELLHOP ray tracing program.

A digitally sampled copy of the transmitted timeseries.
The timeseries transmitted by the source must be available in digital form, sampled at a rate high enough to preclude aliasing effects. This is generally not a problem in the case of software based acoustic modems, since this is exactly what they output. For hardware acoustic modems, one must capture the transmitted timeseries using a general purpose computer equipped with an analog to digital converter and appropriate software.

4 VirTEX for Platform Motion

This section provides a brief tutorial on the use of the *VirTEX for Platform Motion* software. For users that have no previous experience with any of the *VirTEX* family of algorithms, this is a good place to start due to it's relative simplicity. Here are the basic steps involved;

Generate a BELLHOP binary arrivals file for your environment.
The first step in the process is to prepare an input file for the BELLHOP ray tracing program based on the information available for the environment of interest. As with any propagation modeling effort, first running BELLHOP in transmission loss and ray modes can provide valuable feedback

to the user as to the validity of the inputs. Once you are satisfied with your BELLHOP input file, modify it to generate a (binary format) arrivals file, and rerun BELLHOP. Note that multiple receivers located on a regularly spaced grid in range, depth can be modeled if so desired.

Generate the "library" of pre-stretched source waveforms.
The next step in the process is to generate a database or library of pre-stretched versions of the known waveform transmitted by the source by calling the function **delay_sum_lib()**. The source waveform must have a little bit of zero padding both before and after the non-zero envelope of the waveform. This is to insure that the doppler stretched waveforms do not get truncated for the case of receding receivers (where the total duration of the non-zero envelope will increase). The sample rate of the received waveform computed in the next step will match that of the source waveform here, so choose accordingly. The library output by this step is independent of the given environment, so it can be saved (e.g. as a MATLAB save set), and re-used later on without having to recompute it.

Compute the received timeseries by calling the `delay_sum()` function.
The final step in the process is to call the **delay_sum()** function. The primary inputs are the name of the BELLHOP (binary format) arrivals file, the library of pre-stretched source waveforms, and specific values of the source and receiver velocity. The function outputs the timeseries observed at each of the hypothetical receivers defined in the BELLHOP computation.

4.1 VirTEX for Platform Motion - MATLAB Functions

In this section, the specific MATLAB functions that implement the *VirTEX for Platform Motion* algorithm are briefly discussed. A detailed description of the input and output arguments of these functions can be obtained from the MATLAB built-in help system. For example; "**help delay_sum_lib**" will display a detailed description of the usage of that function.

delay_sum_lib.m
The **delay_sum_lib()** function creates a "library" of pre-stretched or "dopplerized" versions of the known waveform transmitted by the source. The caller specifies the minimum and maximum values of the expected receiver velocity, and the number of velocity bins. The output is a self-contained MATLAB structure that can be passed to the **delay_sum()** function. This function internally calls the **arbitrary_alpha()** function, which is located in the `Matlab/Misc` sub-directory of the Acoustics Toolbox distribution.

delay_sum.m
The **delay_sum()** function is the core function of *VirTEX for Platform Motion*. It computes the convolution of the source timeseries with the channel impulse response computed by BELLHOP. The primary inputs are the name of the BELLHOP arrivals file, the structure containing the library of pre-stretched source timeseries, and velocity vectors for both the source and receiver.

4.2 VirTEX for Platform Motion - Example Scripts

In this section, some MATLAB scripts are identified that present some simple examples of how to use the *VirTEX for Platform Motion* functions to solve a practical problem. These scripts are located in the **Examples** sub-directory.

Example_MP.m
The **Example_MP** script presents an example of a fixed source and a grid of receivers, each moving with a specified velocity. For efficiency reasons, it is preferable to compute the "library" of pre-stretched or "dopplerized" versions of the source waveform once, and saving the result as a MATLAB saveset. (The example script recomputes the library each time it is run to make it self-contained). You must run BELLHOP with the **Example_MP.env** input file *before* running this example.

5 VirTEX for Sea Surface Motion

This section provides a brief tutorial on the use of the *VirTEX for Sea Surface Motion* software. Here are the basic steps involved;

Generate a BELLHOP input file for your environment.
The first step in the process is to prepare an input file for the BELLHOP ray tracing program based on the information available for the environment of interest. At the time of this writing, a single invocation of *VirTEX for Sea Surface Motion* can address only *one* receiver. If you wish to model multiple receivers, you must address them one at a time. (The MATLAB utility functions for reading and writing BELLHOP input files can be used to automate that process).

Generate BELLHOP arrivals and eigenray files.
You will need to run BELLHOP twice, once in arrivals mode, and then again in eigenray mode (with everything else being identical for the two runs).

Upsample the source timeseries and compute the Hilbert transform.
The *VirTEX for Sea Surface Motion* algorithm computes the receiver timeseries associated with each eigenray by computing the time that each received sample was transmitted by the source. This computed time is then rounded to the nearest sample of an upsampled copy of the source timeseries (computed by calling the **fourier_upsample()** function and then computing the Hilbert transform of the result). The error introduced by this interpolation can be reduced by upsampling to higher sample rates (at the expensive of additional computational resources).

Compute the received timeseries by calling the delay_sum_surface() function.
The final step in the process is to call the **delay_sum_surface()¿** function. The primary inputs are the name of the BELLHOP (binary format) arrivals file, the name of the BELLHOP eigenray file, the Hilbert transform of upsampled source timeseries, an array of surface heights or handle of a function that can compute surface heights, and velocity vectors for the source and receiver. The function outputs the timeseries observed at the single receiver defined in the BELLHOP computation.

5.1 VirTEX for Sea Surface Motion - MATLAB Functions

In this section, the specific MATLAB functions that implement the *VirTEX for Sea Surface Motion* algorithm are briefly discussed. A detailed description of the input and output arguments of these functions can be obtained from the MATLAB built-in help system. For example; "**help delay_sum_surface**" will display a detailed description of the usage of that function.

delay_sum_surface.m
: The **delay_sum_surface()** function is the core function of *VirTEX for Sea Surface Motion*. The caller must pass the names of the BELLHOP arrival and eigenray files, the Hilbert transform of the transmitted waveform (usually a suitably upsampled copy), values for the source and receiver velocity, and either a matrix of surface heights over appropriate ranges and times or the handle of a function that can be called to compute the height for a given range and time on demand.

fourier_upsample.m
: The **fourier_upsample** function resamples a timeseries to a new sample rate that is a prescribed integer multiple of the original sample rate. It accomplishes this by zero padding in the frequency domain, so the Fourier content is not altered. The error introduced by interpolation in time performed by **delay_sum_surface** can be reduced by passing the transmit timeseries at a sample rate that is at least a few integer multiples of the desired sample rate of the computed receive timeseries.

read_arrivals_bin.m
: The **read_arrivals_bin()** function is called internally by the **delay_sum_surface()** function to read the binary arrivals file produced by BELLHOP. For the typical user, direct calls to this function are not required.

read_rayfil.m
: The **read_rayfil()** function is called internally by the **delay_sum_surface()** function to read the eigenray data file produced by BELLHOP. For the typical user, direct calls to this function are not required.

5.2 VirTEX for Sea Surface Motion - Example Scripts

In this section, some MATLAB scripts are identified that present some simple examples of how to use the *VirTEX for Sea Surface Motion* functions to solve a practical problem. These scripts are located in the **Examples** sub-directory.

Example_Lite_CS.m
: The **Example_Lite_CS** script presents an example of a swell wave that is moving perpendicular (cross swell) to the vertical plane containing the source and receiver. The surface heights are communicated to the **delay_sum_surface()** function by passing it the handle to the **gravity_wave** function that computes the surface wave height on demand for given range and time values. You must run BELLHOP with the **Example_Lite.env** input file *before* running this example, once in arrivals mode, and again in eigenray mode.

Example_Lite_DS.m

The **Example_Lite_DS** script presents an example of a swell wave that is moving parallel (down swell) to the vertical plane containing the source and receiver. The surface heights are passed to the **delay_sum_surface()** function as a matrix of computed values over appropriate range and time values. This example utilizes the WAFO toolbox, which must be installed on your system before you can run this example. Once installed, update your MATLAB paths by running **initwafo('minimal')** located in the top level directory of the WAFO toolbox. You must run BELLHOP with the **Example_Lite.env** input file *before* running this example, once in arrivals mode, and again in eigenray mode.

gravity_wave.m

The **gravity_wave()** function computes the height of a simple (gravity) swell wave for a given range and time. It is called by the **Example_Lite_CS.m** example script described above.

add_noise.m

The **add_noise** script assembles a collection of MATLAB save sets containing simulated receive timeseries, then adds band limited Gaussian white noise. It writes out a single "playback" file (designed for playback to a hardware modem) in signed 16 bit integer format.

6 Full VirTEX

This section provides a brief tutorial on the use of the *Full VirTEX* software. It should be noted that while the underlying concepts behind the *Full VirTEX* algorithm are not that much more complicated than the other variants of the algorithm, the software implementation of the algorithm is *significantly* more complex. In particular, much of the current code is operating system dependent, and only runs on UNIX systems as of this writing.

6.1 Full VirTEX - MATLAB Functions

In this section, the specific MATLAB functions that implement the *Full VirTEX* algorithm are briefly discussed. A detailed description of the input and output arguments of these functions can be obtained from the MATLAB built-in help system. For example; "**help moving_environment**" will display a detailed description of the usage of that function.

Bellhop_execute.m

The **Bellhop_execute()** function is called by the **moving_environment()** function to manage the execution of BELLHOP for *each* time snapshot or frame (one for each time value in the array of surface heights). This function contains code that is operating system dependent and currently works only on UNIX systems. For the typical user, direct calls to this function are not required.

Bellhop_inputs.m

The **Bellhop_execute()** function is called by the **moving_environment()** function to construct BELLHOP input files for *each* time snapshot or frame (one for each time value in the array of

surface heights). This function contains code that is operating system dependent and currently works only on UNIX systems. For the typical user, direct calls to this function are not required.

load_arrfils.m

The **load_arrfils()** function is called by the **moving_environment()** function to read all of the BELLHOP arrival files generated by the call to the **Bellhop_execute()** function. It collates all of the arrival information and returns it as a MATLAB struct. For the typical user, direct calls to this function are not required.

moving_environment.m

The **moving_environment()** function is the core function of *Full VirTEX*. Unlike the other variants of the algorithm, *Full VirTEX* handles the execution of BELLHOP by calling the **Bellhop_inputs()** and **Bellhop_execute()** functions.

6.2 Full VirTEX - Example Scripts

In this section, some MATLAB scripts are identified that present some simple examples of how to use the *Full VirTEX* functions to solve a practical problem. These scripts are located in the **Examples** sub-directory.

Example_Full_DS.m

The **Example_Full_DS** script presents an example of a swell wave that is moving parallel (down swell) to the vertical plane containing the source and receiver. You must edit this script and modify the line where the variable **bparams.bellhop_path** is assigned a value (which specifies where BELLHOP is installed on your system).