Merging VSim's Model Building and Visualization Tools with Custom FDTD Engines

R. Smith [1], A. Weiss [1], R. Bollimuntha [1], S. DMello [1], M. Piket-May [1], M. Hadi [1,2,3], and A. Elsherbeni [3]

[1] Department of Electrical, Computer, and Energy Engineering
University of Colorado, Boulder, CO 80309, USA
ryan.smith-1@colorado.edu, alec.weiss@colorado.edu,
ravi.bollimuntha@colorado.edu,
sanjay.DMello@colorado.edu,
melinda.piket-may@colorado.edu

[2] Department of Electrical Engineering
Kuwait University, Safat 13060, Kuwait
mohammed.hadi@ku.edu.kw

[3] Department of Electrical Engineering & Computer Science
Colorado School of Mines, Golden, CO 80401, USA
aelsherb@mines.edu

Abstract ─ This work demonstrates how the graphical user interface of VSim (electromagnetic simulation software package) is modified and utilized to run a custom finite difference time domain algorithm. Commercial programs typically offer conventional FDTD functionality. More often than not, researchers may want to use their own code versions with proprietary modelling tools and extensions; for example, high-order differencing or specialized absorbing boundary conditions. VSim offers the flexibility of integrating an independent FDTD solver-engine that is tailored for the end user's needs. A detailed example is presented here of the replacement of VSim's own FDTD engine with a high-order FDTD code written with CUDA Fortran. Other custom FDTD codes could be integrated using the presented procedure.

Index Terms ─ CUDA Fortran, FDTD, High Order FDTD methods, VSim.

I. INTRODUCTION

The objective of this paper is to serve as a tutorial for using the graphical user interface (GUI) of VSim [1,2] as input/output interface, initially to define problem parameters and later to visualize the simulation results, while utilizing a custom-made FDTD algorithm. The custom FDTD algorithm used here is the high order FV24 algorithm [3,4], which has an extended unit cell reach that requires modifying some of the standard FDTD simulation parameters. The main constituents of VSim are the *VSimComposer* and *VSim Engine (Vorpal)*. *VSimComposer* is the GUI that allows users, via its Setup page, to define inputs and parameters such as problem space size, material properties and sources.

The *VSimComposer* or *VorpalComposer* also provides Run, Analyze and Visualize functionality. The available options on *VSimComposer* are shown in Fig. 1. Once the structure and behaviour of a simulation model is set using the Setup page, we start the simulation using the Run page. This starts the VSim's own EM computation engine, *Vorpal*, which is based on conventional FDTD. After the simulation is complete, the results, designated electric and magnetic fields, are placed in HDF5 (*.h5*) file format [5] that can be readily visualized using the Visualize page [6]. The main objectives of this work are to show how the same available interface of *VSimComposer* is used to initially set the problem space, import a CAD *.stl* file [6,7], then simulate the problem space by hooking a custom FDTD code to *Vorpal*, so that this custom scheme is run instead of VSim's FDTD engine, and finally to convert the output files to the appropriate (HDF5) format to enable VSim to understand the data to enable visualization. VSim offers the flexibility of integrating an independent FDTD solver-engine that is tailored for the end user's needs by using Python [1,2,8] scripts.

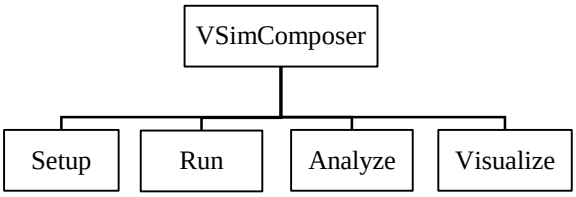

Fig. 1. Available page options on *VSimComposer*.

II. THE SIMLATION INPUT FILE

The input file to any VSim simulation is created with the *.pre* extension [6,9]. An already existing *pre* file can be customized according to the problem. We modify the simulation input parameters such as problem size,

Submitted On: June 3, 2015
Accepted On: January 24, 2016

material properties, scatterers, source location and frequency, etc. These appear in the form of table on Setup page and are collectively called *exposed variables*. We can also view the entire *pre* file and edit from the Setup page as text (using the View Input File button) or it can be modified in any text editor. Scatterers in the form of primitive shapes (rectangular prism, sphere, box, …etc.) or additional wave sources can be included in the problem space by defining their location, sizes, material and frequency (for sources). Scatterers of complex shapes are either formed by combining the primitive shapes or can be imported from a CAD file in binary *.stl* (STL: Stereo Lithography) format [6,9]. Some key variables that are defined in the *pre* file are described in Table 1.

Table 1: Description of key variables defined in *pre* file

Variable Name	Description
Primary_Frequency	The frequency of the source. Any harmonics are defined with respect to this frequency.
X_cells, Y_cells, Z_cells	No. of cells in each direction define the size of the problem.
Cells_ Per_ Wavelength	This gives the resolution of the grid: no. of cells that fit in one wavelength that is calculated from primary frequency.
Timesteps	How long the simulation will run, which also determines how far the wave will propagate.
Dump_Period	How often intermediate results of the simulation will be saved for visualization.
CAD_FILE_NAME	Name of the CAD file example.*stl*)
CAD_MATERIAL	Permittivity of material of scatterers in CAD file (0 for metal, ≥1 for dielectrics)
STRUCTURE_X,Y,Z	Location in the grid where the scatterer in CAD file needs to be placed.

A. Calculations of simulation parameters

Once the *pre* file is constructed as desired, it is validated (button in the upper right hand corner on Setup page) to check the *pre* file for errors. The output window at the bottom gives information about any syntax errors the verifier finds. While the *pre* file is being validated and saved, VSim performs some calculations so that the *.h5* files, placeholders for simulation data, are properly set up to hold all the required EM data.

A few key calculations are given below. The length of a cell is given by:

$$h = \Delta x = \Delta y = \Delta z = \lambda/R, \quad (1)$$

where R is cells per wavelength and,

$$\lambda = c/f, \quad (2)$$

where *f* is the frequency of the source and *c* is the speed of light in free space. The extent in a direction is thus,

$$L_{x,y,z} = hN_{x,y,z}, \quad (3)$$

where $N_{x,y,z}$ is the no. of cells in the x, y or z direction. And the time step is given by:

$$dt = (TIMESTEP\_FACTOR)\, h/(c\sqrt{3}), \quad (4)$$

where *TIMESTEP_FACTOR* is the Courant number.

Various steps involved in the implementation and the *VSimComposer* page they are started from and are given in the form of a flow diagram in Fig. 2.

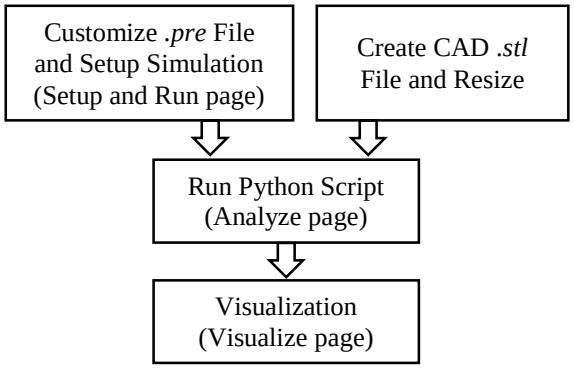

Fig. 2. Various steps involved in the implementation.

III. IMPORTING A CAD FILE

A CAD file is imported by initializing a variable in *pre* file with its name. As of *VorpalComposer 6.0* version, the only types of CAD files available to be imported are *.stl* files [7]. These files are a common format that many CAD programs can export to. If the file is not in this format, currently they are being converted by importing them into a CAD program and then exporting them as a binary *.stl* file. Generally, the *.stl* file has no representation of unit of measure, so the exporting units are arbitrary and need not be changed.

Since *.stl* file does not specify what units its distances are in, it needs to be resized according to the size of the problem space defined in the *pre* file, so that it fits inside the problem space. This task is accomplished by a separate C program (will be posted on ACES website). Currently this program simply scales the files to make each dimension less than a desired size and places the resized file in the working directory.

It is then possible to use the following macro in *pre* file to import shapes from a resized *.stl* file directly:

fillGeoCad(objectName, example_resized.stl,SHAPE_COMPLEMENT,SHAPE_SCALE,SHAPE_TRANSLATION)

By setting *SHAPE_COMPLEMENT* to zero the complement of the shape will not be taken. The *SHAPE_SCALE* is used to scale the resized CAD file shape into meters, the default units for length in *VorpalComposer* [6] for visualization purposes. The *SHAPE_TRANSLATION* can then be used to translate the *.stl* file relative to the location of the origin.

IV. SETTING UP THE SIMULATION

Once the *pre* file is customized as desired, the vorpal engine can be run using Run page on VSimComposer. This accomplishes two tasks: First, it creates empty *.h5* files. The *pre* file is modified such that no actual computations occur here; rather the empty *.h5* files are created to serve as placeholders for EM data (to be computed by the custom FV24 engine in the next step). This implies that the EM value for each cell at each dump time is temporarily 0. This is accomplished by using *dummyUpdater* in the *FieldUpdater* block in the *pre* file [9].

Second, it converts the resized *.stl* file to *.h5* file. The *pre* file is also modified to capture any resized *.stl* file placed in the working directory. It gets converted to *.h5* file that contains tables of information about the scatterer used for VSim's visualization. One of the tables contains the coordinates of the locations the scatterer occupies in the grid. This table is later picked and converted to .txt file to be used by the custom FDTD engine.

V. RUNNING THE SIMULATION

Now that the simulation has been set up, the custom FDTD engine, is called next. This is done using the Analyze page and selecting the Python script copied earlier to the working directory. This script can also be run from the command prompt after navigating to the working directory. The following subsections give a description of what this script does.

A. Parse the *pre* file

This is a string search that finds the names of variables and grabs their values. They are saved as python variables.

B. Convert scatterer *.h5* file into text file

Converts one of the tables in the scatterer *.h5* file generated earlier into a *.txt* file. This text file contains the coordinates of the scatterer elements in the grid. Material properties of the scatterer specified in the *pre* file are also added to the text file in the conversion process. This text file is later used by the custom FDTD engine.

C. Construct object arrays

An array of integers is constructed for each primitive object and additional source defined in *pre* file. This array identifies whether the object is sphere, box, etc. or a source and contains information on location and size.

D. Make a call to the custom FDTD engine

The compiled and ready-to-run custom FDTD engine is called and also, the parameters grabbed from the *pre* file and object arrays are passed to the custom FDTD engine. This engine should also pick the text file containing scatterer and material data. For our example, the results of the execution are saved in *.csv* (comma separated values) files as *Dump_1E.csv* (for electric field), *Dump_2E.csv*, etc. based on the dump period specified in *pre* file. There will be one electric field file and one magnetic field file for each data dump.

E. Conversion from *.csv* to *.h5* data format

Another python script is called that takes the data in the *.csv* files generated by custom code, and populates the *.h5* placeholders that were created by Vorpal engine earlier in the working directory.

VI. DATA VISUALIZATION

The EM data calculated by the custom FDTD engine are now present in the *.h5* files in the working directory, and can be viewed using the Visualize page of *VSimComposer*. VSim automatically detects the *.h5* files. Here is a list of important features in the 3D visualizer: "Display Contours" box in the lower left corner needs to be checked to see the wave nature of fields. On the left side, there are drop-down menus for Scalar and Vector data, as well as geometric objects. In the Scalar tab, the x, y, z component or the magnitude of either the electric or magnetic field can be selected. In the vector field tab, the electric or magnetic vector field can be selected. In the Geometries section, the user can check the primitive objects and imported CAD model to view them.

Once the data loads, the 3D visualizer can be used to pan and zoom around the waves. The time step bar at the bottom of the window is used to move in time. The number of contours shown can be changed between 2 and 20; this can be used to display every variation in the field, or only the major ones. Using the Colors button, the color scale of the visualizer can be changed. By default, the scale will update for each time step. If one wishes to keep the scale set for all time steps, the minimum and maximum values in the color options dialog may be fixed. This is useful for observing how the field strength decreases over time. A 3D view of electric field magnitude and the scatterer is shown in Fig. 3. In

addition to 3D visualization, 2D cuts can be made to observe waves in 2D space using the Data View menu. A 2D view of electric field and the scatterer is shown in Fig. 4.

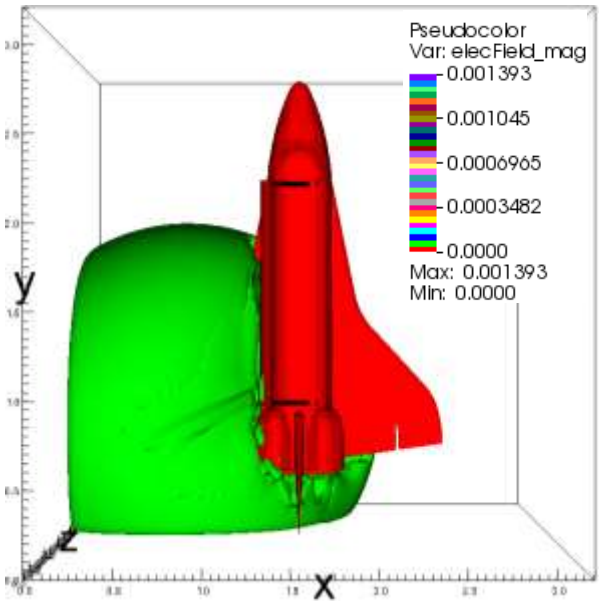

Fig. 3. A 3D view of electric field magnitude and shuttle scatterer at t = 180 time steps.

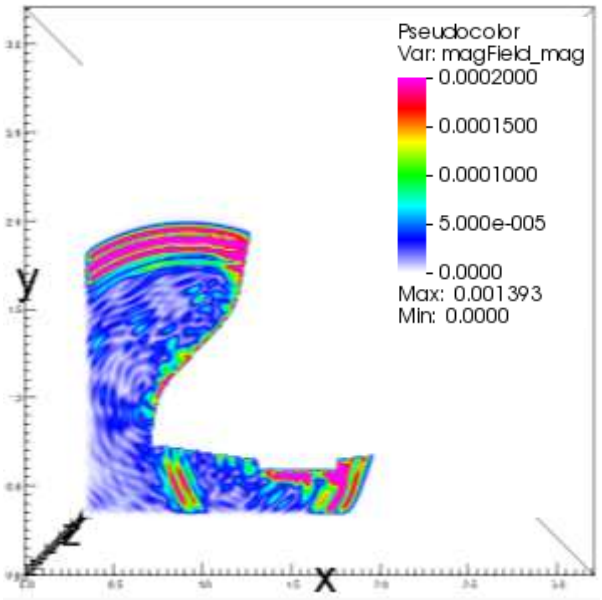

Fig. 4. A 2D slice in XY plane showing magnetic field magnitude at t = 180 time steps.

VII. SUMMARY AND CONCLUSION

A step-by-step implementation of a custom FDTD engine using the GUI data input and data visualization modules of a commercial EM simulation package is presented. Problem space defining parameters are set using the data input tool. Embedded scatterer definitions, if simple enough, could be defined by the same tool, or imported as a CAD file. The CAD file is then reshaped and combined with the user provided custom FDTD engine using a Python script. The visualization tool of the software package is finally used to observe and analyze collected simulation data. Although a custom high order FDTD engine was used as an example here, readers are at liberty to use their own FDTD engines written in their programming language of choice. A key data reshaping software tool will be posted to ACES website to facilitate this process.

ACKNOWLEDGEMENT

The authors would like to thank Tech-X Corp. for providing the license and support for the commercial software used for this work: VSim-7.0.0.

REFERENCES

[1] *VSim and VSim for Electromagnetics*, [Online]. Available: http://www.txcorp.com

[2] M. Piket-May, S. DMello, R. Smith and M. Hadi, "Using the VSim GUI to visualize high-order FV24 simulations of electrically large systems," *IEEE Antennas Propagat. Society Int. Symp. (APSURSI)*, Memphis, TN, pp. 1632-1633, July 2014.

[3] M. F. Hadi, "A finite volumes-based 3-D low dispersion FDTD algorithm," *IEEE Trans. Antennas Propagat.*, vol. 55, no. 8, pp. 2287-2293, Aug. 2007.

[4] M. F. Hadi, "CUDA Fortran acceleration for the finite-difference time-domain method," *CPC*, vol. 184, no. 5, pp. 1395-1400, Jan. 2013.

[5] *What is HDF5?*, [Online]. Available: https://www.hdfgroup. org/HDF5

[6] *VSim in Depth*, Release 7.0.0, Tech-X Corporation, Boulder, CO, July 2014.

[7] *The STL Library*, [Online]. Available: http://www.eng.nus.edu.sg/LCEL/RP/u21/wwwroot/stl_library.htm

[8] *Beginner's Guide to Python*, [Online]. Available: https://wiki.python.org/moin/BeginnersGuide

[9] *VSim Reference Manual*, Release 7.0.0, Tech-X Corporation, Boulder, CO, July 2014.