2014 CMBEC37 Conference
Vancouver, BC
May 21 – 23, 2014

MODELING OF BIOELECTROMAGNETIC EFFECTS USING THE TRANSMISSION LINE MATRIX METHOD

Poman P.M. So
Department of Electrical and Computer Engineering
University of Victoria

INTRODUCTION

The electrical properties of human tissues affect the interaction of electromagnetic (EM) fields with the human body. The effects depend on the tissue properties as well as on the strength and frequency of the EM signals. These effects can be modeled using numerical methods. This paper describes the essence of a graphic processing unit accelerated Transmission Line Matrix (TLM) method with emphasis on applying TLM to bioelectromagnetic modeling.

THE TRANSMISSION LINE MATRIX METHOD

The Transmission Line Matrix (TLM) is a computational algorithm based on the Huygens Principle which stipulates that a wavefront has of a large number of radiators which give rise to spherical wavelets. These wavelets interact with each other to form a new wavefront [1]. This idea is depicted in the Treatise of Light using a candle's flame as shown in Figure 1. The systems of wavelet due to sources A, B, and C, respectively, propagate away from the flame and combine with each other to form a new wavefront. The point sources on the new wavefront in turn give rise to another generation of wavelets. This recursive process can be implemented using a mesh of transmission lines as shown in Figure 2.

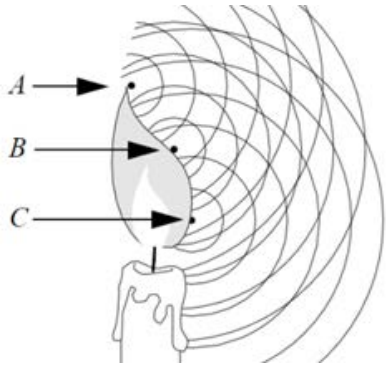

Figure 1: Illustration of Huygens' Principle. A candle and three point sources (A, B, and C) on the surface of the flame. Each point source generates a family of circular wavelet. The concentric wavelets generated by the point sources propagate radially away from the center. Circles having the same radius represent wavelets having the same phase shift with respect to their source location.

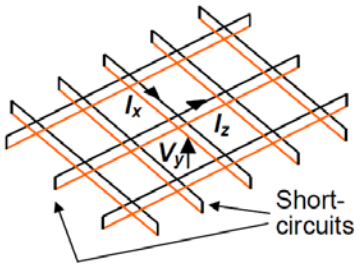

Figure 2: A mesh of transmission lines. The voltage impulse scattering and transferring procedure (equations 2 and 3) can be used to model the wave propagation behavior described by Huygens' Principle.

As the space between transmission-lines decreases, the wave property of the mesh approaches to that of the continuum space which the mesh represents. In order to implement the transmission line mesh, hence the Huygens' wave model, on a digital computer, one must formulate the algorithm in the discretized form; i.e. both space and time must be represented in terms of finite elementary units, Δl and Δt, which are related by the velocity of light, c, such that:

$$\Delta l = c \times \Delta t \qquad (1)$$

Two-dimensional space can then be modeled by a Cartesian matrix of points, called nodes, separated by a distance Δl. The unit time it takes for electromagnetic signal to travel from one node to the next is Δt. The fundamental procedures of the TLM method are shown in Figure 3.

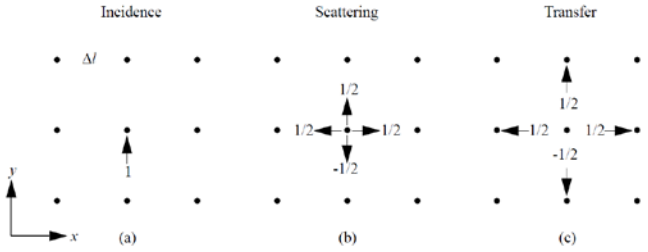

Figure 3: Huygens' wave model in the discretized two-dimensional space. (a) Incidence of a short voltage impulse at a space point (scattering center). (b) Scattering of the impulse. (c) Transfer of the scattered impulses to neighboring nodes.

According to the transmission line theory a voltage impulse incident on a transmission line junction will have its energy scattered in all directions. If such an impulse is incident on one of these junctions, also called nodes, in a TLM mesh from the negative y-direction, each scattered impulse must carry one fourth of the energy on the incident impulse. The scattered field quantities must then be 1/2 in magnitude. Furthermore, the reflection coefficient seen by the incident impulse must be –1/2 to ensure field continuity at the node. This event can be described by the following matrix equation:

$$ {}_{k+1}\boldsymbol{V}^{r} = \boldsymbol{S} \times {}_{k}\boldsymbol{V}^{i} \tag{2} $$

where $\boldsymbol{V} = \begin{bmatrix} V_1 \\ V_2 \\ V_3 \\ V_4 \end{bmatrix}$, $\boldsymbol{S} = \frac{1}{2}\begin{bmatrix} -1 & 1 & 1 & 1 \\ 1 & -1 & 1 & 1 \\ 1 & 1 & -1 & 1 \\ 1 & 1 & 1 & -1 \end{bmatrix}$, V_1, V_2, V_3, and V_4 represent the voltage impulses incident from the bottom $(-\text{ve}\, y)$, left $(-\text{ve}\, x)$, top $(+\text{ve}\, y)$, and right $(+\text{ve}\, x)$ of a node. The superscripts i and r denote incident and reflected impulses. The subscripts k and $k+1$ represent the time in Δt. Furthermore, any impulse emerging from a node becomes automatically an incident impulse on their neighboring node (Figure 3b and c). This event can be best described by the following set of equations:

$$ \begin{aligned} {}_{k+1}V_1^i(x,y) &= {}_{k+1}V_3^r(x,y-1) \\ {}_{k+1}V_2^i(x,y) &= {}_{k+1}V_4^r(x-1,y) \\ {}_{k+1}V_3^i(x,y) &= {}_{k+1}V_1^r(x,y+1) \\ {}_{k+1}V_4^i(x,y) &= {}_{k+1}V_2^r(x+1,y) \end{aligned} \tag{3} $$

The coordinates in (3) are normalized to the space discretization unit, Δl. Equations (2) and (3) together form the basic algorithm of the TLM method. Thus, if the magnitudes, positions and directions of all voltage impulses are known at time k, then the corresponding values of the voltage impulses at time $k+1$ can be obtained by operating equations (2) and (3) on each node in the mesh.

MODELING OF MATERIAL AND BOUNDARY PROPERTIES

To model biological tissues, transmission line stubs of various electrical properties and terminations are inserted at the centered of the nodes. For impulse synchronism reason, the length of these loading stubs must be one-half of the mesh spatial resolution Δl. As long as the mesh parameters Δl is small when compared with the wavelength of interest, the voltage and current relationship in the x- and y-direction of the TLM mesh can be represented by the following differential equations [1]:

$$ \begin{aligned} \frac{\partial V_z}{\partial x} &= -L\frac{\partial I_x}{\partial t} \\ \frac{\partial V_z}{\partial y} &= -L\frac{\partial I_y}{\partial t} \\ \frac{\partial I_x}{\partial x} + \frac{\partial I_y}{\partial y} &= 2C\left(1 + \frac{y_o}{4}\right)\frac{\partial V_z}{\partial t} - \frac{g_o\sqrt{C/L}}{\Delta l}V_z \end{aligned} \tag{4} $$

The corresponding Maxwell's equations in a lossy medium obtained by setting $dy/dz = 0$ and $E_x = E_y = H_z = 0$ are:

$$ \begin{aligned} \frac{\partial E_z}{\partial x} &= \mu\frac{\partial H_y}{\partial t} \\ \frac{\partial E_z}{\partial y} &= -\mu\frac{\partial H_x}{\partial t} \\ \frac{\partial H_y}{\partial x} - \frac{\partial H_x}{\partial y} &= \epsilon\frac{\partial E_z}{\partial t} + \sigma E_z \end{aligned} \tag{5} $$

The following equivalences can be established by comparing equations (4) and (5):

$$ \begin{aligned} &E_z \equiv V_z \qquad H_x \equiv I_y \qquad H_y \equiv -I_x \\ &\epsilon \equiv \epsilon_o\epsilon_r \qquad \epsilon_o \equiv 2C \qquad \epsilon_r \equiv 1 + \frac{y_o}{4} \\ &\mu \equiv L \qquad \sigma \equiv \frac{g_o\sqrt{C/L}}{\Delta l} \end{aligned} \tag{6} $$

The shape and size of tissues in various organs can be defined by setting the stub values accordingly. Since the size of the TLM computation domain must agree with the biological model which in turn is limited by the amount of available computer memory, conducting and absorbing boundaries are used to define the modeling space. These boundaries can be implemented by introducing appropriate impulse reflection coefficients, $\Gamma_{impulse}$, in the TLM mesh. The reflection coefficients must be:

- real numbers because TLM is a time-domain procedure,
- implemented at the center of a node or halfway between two nodes because the movement of the voltage impulses in the mesh must be synchronized,
- chosen such that the interaction of the electromagnetic wave, represented by the superposition of impulses in the mesh, with the boundary is properly modeled.

The second condition implies that curved boundaries must be modeled by piecewise straight sub-boundaries. The impulse reflection coefficient for perfect electric and magnetic boundaries are -1 and 1, respectively. For simple absorbing boundaries, the reflection coefficient depends on the angle of incidence, θ:

$$ \Gamma_{impulse}(\theta) = \frac{1-\sqrt{2\epsilon_r\cos(\theta)}}{1+\sqrt{2\epsilon_r\cos(\theta)}} \tag{7} $$

A thorough discussion of various types of boundaries, lossy material, and the generalized three dimensional implementation of the TLM method are given in [1] to [3].

MODELLING OF BIOELECTROMAGNETIC EFFECTS

In order to analyze a complete human body with the 3.6 to 2.0 mm resolution data models, the TLM mesh will have 5 to 30 mega voxels, figure 4. These resolutions are good enough for modeling of human tissue interacting with microwave frequency signal released by common wireless devices such as cellphones and routers. Using equation (1), with $\epsilon_r = 30$ [6] and $\Delta\ell = 2$ mm, the modeling time step between two successive updates is 36 picoseconds. The period of a 3-GHz microwave signal is about 33 picoseconds. Hence, the computation time step is in the same order of the signal's temporal resolution. A smaller modelling resolution, $\Delta\ell$ may be used if more accurate result is needed but that would reduce the modeling time step and increase the overall numerical effort.

With the above estimation, to iterate over a 30 mega-voxel human model once using the TLM method would require more than 3 billion floating point operations. Running such a large number of TLM operations hundreds of thousands of times on traditional CPU hardware would require days of CPU times. Thanks to the advancement in graphic processing unit (GPU) technology, it is now possible to bring supercomputing power to the PC world. Numerical procedures such as TLM can be accelerated by two orders of magnitudes by using desktop and laptop PCs equipped with a GPU. Simulating a TLM mesh on these PCs is a two-step process: 1) Allocate mesh memory on the host computer's CPU, and 2) map the TLM mesh on the CPU to the GPU using a GPU computing library. In this paper, we will use Microsoft's C++ AMP library. The two required steps are illustrated in Listings 1 and 2. With the mesh memory allocated on the GPU, the TLM impulse scattering and transfer procedures with inhomogeneous material can be executed in parallel using the C++ AMP code shown in Listing-3 and 4, respectively.

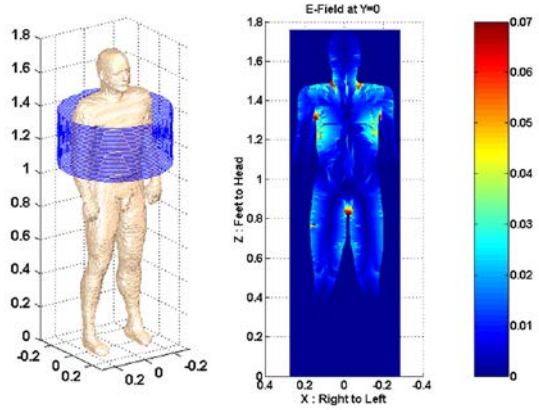

Figure 4: A 2mm resolution human-model in a gradient coil for peripheral nerve simulation [4, 5].

Listing-5 shown the C++AMP implementation of the TLM boundary operations. The code segments in Listings 3 to 5 constitute the main operations of the TLM algorithm. The codes are executed repeatedly inside an iteration loop until the maximum number of iterations is reached. Finally, data is transferred back to the CPU by the code in Listing-6.

```
// Define storage on the host computer, i.e. CPU
//=====================================
pLeft   = new std::vector<float>(size_of_mesh), // v1
pTop    = new std::vector<float>(size_of_mesh), // v2
pRight  = new std::vector<float>(size_of_mesh), // v3
pBottom = new std::vector<float>(size_of_mesh), // v4
pStub   = new std::vector<float>(size_of_mesh), // v5
pYo     = new std::vector<float>(size_of_mesh),
pGo     = new std::vector<float>(size_of_mesh);
```

Listing-1: C++ code segment for allocating a two-dimensional TLM mesh on the CPU.

```
// Define storage on the client processors, i.e. GPU
//=======================================
extent<2> ex(XSize,YSize);
ap_left  = new array<float, 2>(ex, pLeft->begin());
ap_top   = new array<float, 2>(ex, pTop->begin());
ap_right = new array<float, 2>(ex, pRight->begin());
ap_bottom= new array<float, 2>(ex, pBottom->begin());
ap_stub  = new array<float, 2>(ex, pStub->begin());
ap_yo    = new array<float, 2>(ex, pYo->begin());
ap_go    = new array<float, 2>(ex, pGo->begin());
```

Listing-2: Code segment for mapping the two-dimensional TLM mesh created in Listing-1 on to the GPU. The extent and array are C++ AMP library objects for GPU computing.

```
// Define reference variables
//=====================
array<float, 2>
 &a_top(*ap_top),   &a_bottom(*ap_bottom),
 &a_left(*ap_left), &a_right(*ap_right),
 &a_stub(*ap_stub), &a_yo(*ap_yo),  &a_go(*ap_go);

// C++ AMP implementation of impulse scattering
//=======================================
parallel_for_each(a_top.extent, [&](index<2> ij) restrict(amp)
{
  float yo = a_yo[ij];
  float va = 2.0f/(4.0f+a_go[ij]+yo);
    float vb = (a_top[ij]+a_left[ij]+a_bottom[ij]
          +a_right[ij]+yo*a_stub[ij])*va;
  a_top[ij]   = vb - a_top[ij];
  a_left[ij]  = vb - a_left[ij];
  a_bottom[ij] = vb - a_bottom[ij];
  a_right[ij] = vb - a_right[ij];
  a_stub[ij]  = vb - a_stub[ij];
});
```

Listing-3: TLM scattering procedure in C++ AMP.

```
//=====================================
// C++ AMP implementation of impulse transfer
//=====================================
parallel_for_each(a_top.extent, [&,M,N](index<2> ij) restrict(amp)
{
   index<2> ip1(ij[0]+1, ij[1]), jp1(ij[0], ij[1]+1);
   if (ij[0]<M) { float temp = a_right[ij];
                  a_right[ij] = a_left[ip1];
                  a_left[ip1] = temp;
   }
   if (ij[1]<N) { float temp = a_bottom[ij];
                  a_bottom[ij] = a_top[jp1];
                  a_top[jp1]   = temp;
   }
});
```

Listing-4: TLM impulse transfer procedure in C++ AMP.

```
//=============================================
// C++ AMP implementation of simple boundary operation
//=============================================
parallel_for_each(a_top.extent, [&,M,N](index<2> ij) restrict(amp)
{
   index<2> ip1(ij[0]+1, ij[1]), jp1(ij[0], ij[1]+1);

   if (ij[0]<M && a_hRefl[ij]!=NO_BOUNDARY) {
      float temp = a_right[ij] * a_hRefl[ij];
      a_right[ij] = a_left[ip1] * a_hRefl[ij];
      a_left[ip1] = temp; }
   if (ij[1]<N && a_vRefl[ij]!=NO_BOUNDARY) {
      float temp = a_bottom[ij] * a_vRefl[ij];
      a_bottom[ij] = a_top[jp1] * a_vRefl[ij];
      a_top[jp1]   = temp; }
});
```

Listing-5: TLM impulse reflection operations in C++ AMP.

```
//=======================
// Transfer data on GPU to CPU
//=======================
extent<2> ex(XSize,YSize);
array_view<float,2> av_top(ex,*pTop);
array_view<float,2> av_left(ex,*pLeft);
array_view<float,2> av_bottom(ex,*pBottom);
array_view<float,2> av_right(ex,*pRight);
array_view<float,2> av_stub(ex,*pStub);
ap_top->copy_to(av_top);
ap_left->copy_to(av_left);
ap_bottom->copy_to(av_bottom);
ap_right->copy_to(av_right);
ap_stub->copy_to(av_stub);
```

Listing-6: Code segment for transferring TLM data on the GPU back to the CPU.

ALGORITHM PERFORMANCE

The C++ AMP TLM procedures described in this paper have been incorporated into a TLM electromagnetic field simulator. The C++ AMP accelerated parallel TLM algorithms can run at a top speed of 83 Gflops/second on a NVIDIA Tesla C2050 GPU. The performance can be further enhanced by using some advanced C++ AMP memory tiling features. A discussion on this advanced programing technique is beyond the scope of this paper. Readers interested in this technique are referred to Microsoft's online documentation on C++ AMP, [7].

CONCLUSION

An accelerated massively parallel TLM engine has been incorporated into a TLM simulator. The software can run at a top speed of 83 Gflops/second on the NVIDIA Tesla C2050 GPU which is two orders of magnitude faster than an equivalent CPU version running on an Intel i7 processor. The performance can be further enhanced by using C++ AMP memory tiling controls. A graphical user interface for biomedical modeling of human model and multi-physics capabilities are being developed for this GPU accelerated TLM solver.

REFERENCES

[1] W.J.R. Hoefer, "The transmission-line matrix method — theory and applications", IEEE Trans. on Microwave Theory and Techniques, vol. MTT-33, no.10, pp. 882–893, October 1985.

[2] F.V. Rossi, P.P.M. So, N. Fichtner and P. Russer, "Massively Parallel Two-Dimensional TLM Algorithm on Graphics Processing Units", pp. 153 – 156, IEEE MTT-S International Microwave Symposium, Atlanta, Georgia, June 15 – 20, 2008.

[3] P.P.M. So, "Numerical modeling of electromagnetic structures with TLM on NVIDIA graphics processors", pp.885–887, IEEE Asia Pacific EMC Symposium, Beijing, 12–16 April 2010.

[4] P.P.M. So, K. Caputa, and M.A. Stuchly, "Peripheral nerve stimulation by gradient switching fields in MRI", Proceedings of the 25th Annual International Conference of the IEEE Engineering in Medicine and Biology Society, Cancun, Mexico, pp. 3771–3774, 17-21 September 2003.

[5] P.P.M. So, M.A. Stuchly, and J.A. Nyenhuis, "Peripheral nerve stimulation by gradient switching fields in magnetic resonance imaging", IEEE Trans. Biomed. Eng, Vol. 51, No. 11, pp. 1907–1914, Nov. 2004.

[6] C. Furse, D.A. Christensen, and C.H. Duney, "Basic Introduction to Bioelectromagnetics", Table A1, pp. 254, CRC Press, 2009.

[7] Microsoft C++ AMP documentation, http://msdn.microsoft.com/en-us/library/hh265137.aspx, {2014-02}