Chapter 2

# Post Processing of Results of EM Field Simulators

Tomas Vydra and Daniel Havelka

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/46452

## 1. Introduction

In this chapter we shall focus on the needs that many researchers, scientists and even students have very often. When using commercial simulation software for numerical simulation of electromagnetic field we frequently encounter many insufficiencies which those software products have. Usually, main aim of computational software developers is to optimize and refine so called core of these programmes – EM field solver. After that CAD (Computer Assisted Design) and post processing parts of EM simulators are dealt with. Mainly this can be an issue with newer, short-time in development products but one's own post processing using Matlab can be greatly beneficial even when using well established simulators of EM field. This is largely due to its flexibility which cannot be overcome by any EM field simulator.

Throughout this chapter we will show you many ways of post processing which we use in our research of EM field in industrial and medical applications (Vrba et al., 2008; Vydra et al., 2011) and in primary research of EM field around living cells and structures inside cell bodies (Cifra et al., 2011; Havelka et al., 2011).

We hope that this chapter will aid many researchers and students in the vast field of EM research. Here, we present our knowledge and tips which we have gathered through our studies and our research.

### 1.1. Technical introduction

Generally, rough results we obtain using simulators of electromagnetic field - or from analytical solution of systems described by discrete elements - are in the form of complex vector components of intensity of electric and magnetic field (i.e. time dependent – periodical – components in the directions of coordinate axes). We process these results using Matlab and interpret them to draw conclusions. In this chapter we would like to present basic processing of rough data, calculation of specific absorption rate and other parameters

in particular regions of simulation domain, visualization of results in many ways (pcolor, slices, histograms, multiple iso-surface, surf interpretation on various shapes according to specific task etc.). We will provide detailed examples with practical applications and explanation of advantages provided by presented solutions.

## 2. Rough results from EM field simulator

As mentioned above, in this chapter we suppose that we have obtained rough data from any numerical simulator of EM field and now we want to interpret them. First of all we should look at how the structure of this data looks like. To get the full understanding we shall briefly go through some EM field basics.

### 2.1. EM field basics

Electromagnetic Field can be described using well known Maxwell's equations (for more information on Maxwell's equations please refer to any book dealing with EM field theory).

$$\left.\begin{aligned} \oiint D dS &= Q \\ \oiint B dS &= 0 \\ \oint E dl &= -\frac{d\phi}{dt} \\ \oint H dl &= I_0 + I_c \end{aligned}\right\} \tag{1}$$

Simply by solving those equations EM field can be completely described at all points of space and time. This leads us to complete description of EM field using only phasors of intensity of electric and magnetic field $\boldsymbol{E}$ and $\boldsymbol{H}$ (or $\boldsymbol{D}$ and $\boldsymbol{B}$ where $\boldsymbol{D} = \varepsilon\boldsymbol{E}$ and $\boldsymbol{B} = \mu\boldsymbol{H}$). This means that output of conventional commercial simulator is in the form of time dependent vectors that have components in axis $x$, $y$ and $z$. These vectors are defined for each part of computational domain (e.g. when using FDTD (Thomas et. al., 1994), vectors are defined for each voxel – block discretizing computational domain).

We can see that this type of data can be extracted in form of matrices (multi-dimensional, e.g. 4D). Now, we shall look closer at those matrices.

### 2.2. Data structure

As we mentioned in previous chapter, results from simulators of EM field are represented as matrices, which directly predestines them to be processed in Matlab, which is the perfect tool for matrix operations.

There is a sample of data obtained from simulation in the following table. It depicts x-component of vector of intensity of electric field [V/m] in y-axis section in a part of some model.

| X/Z [mm] | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| **1** | 10 | 10 | 11 | 12 | 13 | 14 | 12 | 9 | 8 | 7 |
| **2** | 10 | 10 | 11 | 12 | 13 | 14 | 11 | 10 | 9 | 8 |
| **3** | 11 | 11 | 11 | 12 | 13 | 14 | 13 | 11 | 10 | 9 |
| **4** | 12 | 12 | 12 | 12 | 13 | 14 | 13 | 12 | 11 | 10 |
| **5** | 13 | 13 | 13 | 12 | 13 | 14 | 12 | 12 | 12 | 11 |
| **6** | 14 | 14 | 13 | 13 | 13 | 14 | 13 | 13 | 13 | 12 |
| **7** | 13 | 14 | 13 | 13 | 13 | 14 | 14 | 14 | 14 | 12 |
| **8** | 13 | 14 | 14 | 14 | 14 | 14 | 15 | 15 | 14 | 13 |
| **9** | 13 | 13 | 13 | 14 | 15 | 15 | 16 | 15 | 14 | 13 |
| **10** | 12 | 13 | 13 | 14 | 14 | 15 | 15 | 15 | 14 | 13 |

**Table 1.** X-component of vector of intensity of electric field [V/m]

Following graphical representation can help us shed some more light on the structure of data we obtained. These data are represented as four dimensional matrices (for phasors ***E*** and ***H*** separately) depicting whole computational domain and they are time dependent.

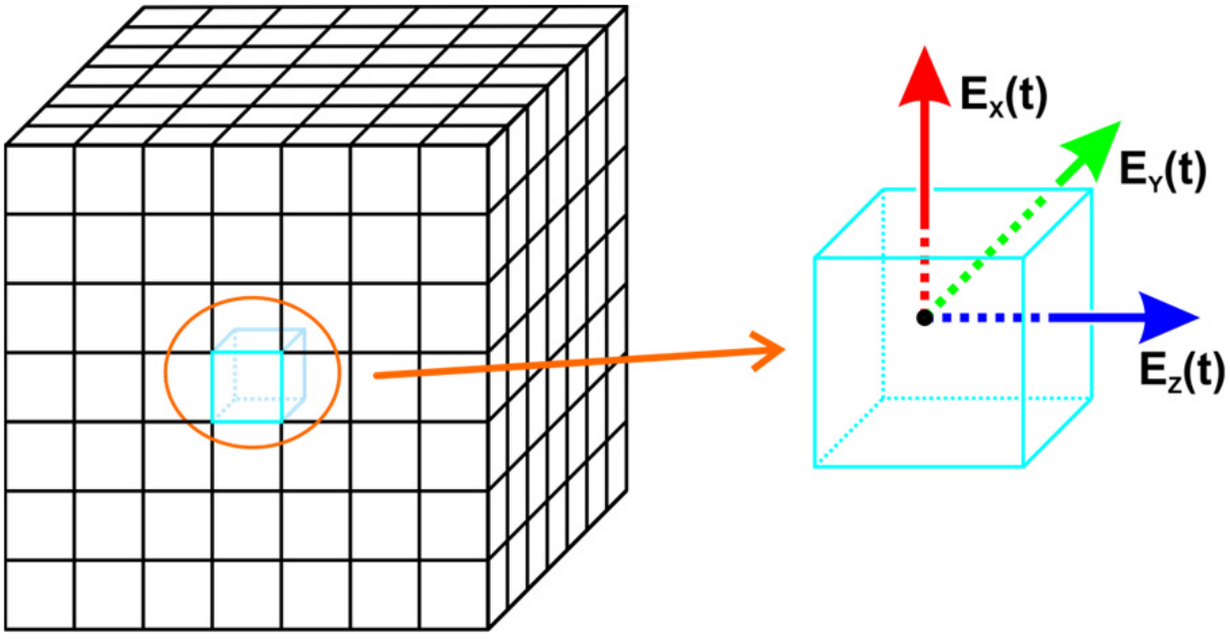



**Figure 1.** Data structure

Generally we can describe phasors as follows.

$$\widehat{V}_{(x,y,z,t)} = V_{(x,y,z)} e^{j\omega t} \tag{2}$$

Note: It may be necessary to convert data to suitable matrix form (e.g. rough data are in the form of a row vector with axial information for each element). We will look into it in the chapter 3.

Now that we know what our data source looks like we can simply process it to view the results and highlight some of their aspects according to our needs (see Table 2. for axial information).

```
73 -    actualAxisX(1) = loadedVectorX(1);
74 -    for a = (2:length(loadedVectorX))
75 -        if loadedVectorX(a) == loadedVectorX(1)
76 -            break;
77 -        else
78 -            actualAxis(a) = loadedVector(a);
79 -        end;
80 -    end;
81
82 -    d = 2;
83 -    actualAxisY(1) = loadedVectorY(1);
84 -    for b = (2:length(loadedVectorY)-1)
85 -        if loadedVectorY(b+1) ~= loadedVectorY(b)
86 -            actualAxis(d) = loadedVector(b+1);
87 -            d = d + 1;
88 -        else
89 -            if loadedVector(b+1) == loadedVector(1)
90 -                break;
91 -            end;
92 -        end;
93 -    end;
94
95 -    e = 2;
96 -    actualAxisZ(1) = loadedVectorZ(1);
97 -    for c = (2:(length(loadedVectorZ)-1))
98 -        if loadedVectorZ(c+1) ~= loadedVectorZ(c)
99 -            actualAxisZ(e) = loadedVectorZ(c+1);
100 -            e = e + 1;
101 -        end;
102     end;
```

**Figure 2.** Extraction of Axes (in our example)

## 3. Viewing the results

In this section we are going to show some examples of how obtained data can be viewed, how to interpret those results, what type of projection should we use etc. We shall illustrate this on some practical examples of EM field applications.

### 3.1. Basic transformation of rough data

As mentioned above we might obtain rough data in the form of a row vector. Let us illustrate this in this simple example. Our computational domain is 2 by 2 by 2 thus obtained row vector (x-component, apmlitude) has 8 elements. See Table 2.

| **x axis** | 1 | 2 | 1 | 2 | 1 | 2 | 1 | 2 |
|---|---|---|---|---|---|---|---|---|
| **y axis** | 1 | 1 | 2 | 2 | 1 | 1 | 2 | 2 |
| **z axis** | 1 | 1 | 1 | 1 | 2 | 2 | 2 | 2 |
| **Vector of values** | 8 | 6 | 5 | 2 | 5 | 6 | 8 | 9 |

**Table 2.** Amplitude of x-component of intensity of electric field [V/m]

From this we can extract axis. As long as we do not know the length of each axis we need to utilize this method to find actual axial data (see Figure 2.).

Note that actual axis arrangement can be different in your case (e.g. x and y axis arrangement may be commuted). Thus it is vital to get familiar with axial arrangement in your exporter from EM field simulator.

Furthermore, `actualAxis` vectors are underlined because they are growing on every loop iteration. Since axial vectors are not usually very long, this poses only mild concern. They cannot be preallocated because we generally do not know their actual length. If you are expecting very long axial vectors you may consider preallocating them safely longer than your expectations and then just using part of them which is non-zero (you may select this part of a vector using `find` – please refer to Matlab documentation).

Now that we know the length of each axial vector we may sort the vector of exported values and transform it to a matrix which will better represent three dimensional nature of our computational domain and will allow us to plot data with axial information. In this case we can very well utilize `reshape` which is built in Matlab.

MATRIX = reshape(vector_of_values,lenght(X),lenght(Y),lenght(Z))

Thanks to this process we now have every component of each vector (i.e. $\boldsymbol{E}$ and $\boldsymbol{H}$) represented as three dimensional matrix and we can utilize it further.

## 3.2. Basic plotting of data

First of all, we need to bear in mind that we have time-dependent data. The most basic process is to plot actual situation (distribution of intensity of electric or magnetic) at a given time, or amplitude of vector. (In some applications we may need to plot just one component of this vector. This is even simpler because then we can disregard following method.)

Phasor of intensity of electric or magnetic field can be represented by modulus and phase or real and imaginary part. We need to merge all the components of the vector and obtain real and imaginary part. This can be simply done (i.e. vector adding component matrices together). Then we have one matrix of complex numbers. We can choose specific time in which we need EM field to be plotted simply by adding $\langle 0,2\pi \rangle$ to the phase of each vector and then we can plot real and imaginary modulus of the vector in specified time. Or we take just the amplitude of vectors and plot them.

It is very usual to plot RMS (i.e. Root Mean Square) value of vectors which is defined as follows.

$$RMS|E| = \frac{|E|}{\sqrt{2}} \tag{3}$$

This can be again obtained very simply from amplitude of intensity of electric field. (Note that this same procedure can be used also in the case of intensity of magnetic field ***H***, usually in applications involving heating and/or drying we deal only with intensity of electric field ***E***, because it is the source of heat generation in exposed samples.)

Now that we have three dimensional matrix of values of $RMS|E|$ we can plot it to see what our results look like. In the following section there is an example we prepared to illustrate how the results can be viewed and interpreted.

### *3.2.1. Example of basic data plotting*

In this section we shall extract data from a simulation which has setup according to the Figure 3. Please note that this is just an example without any practical use, it serves only as an illustration.

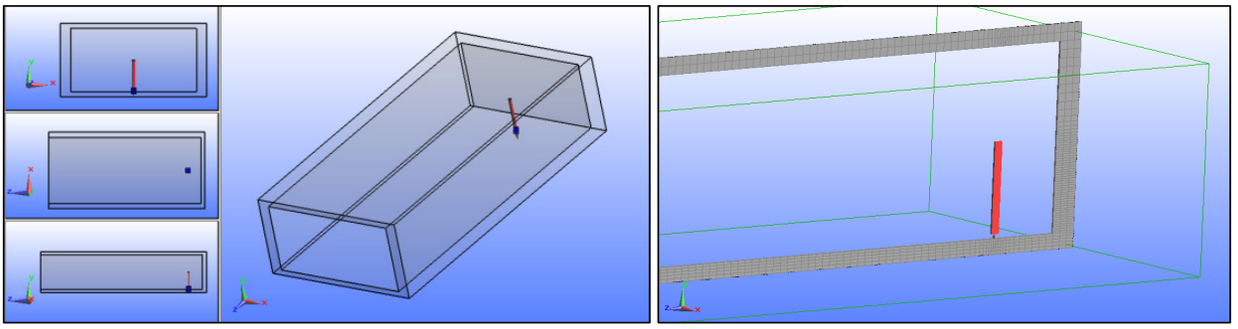
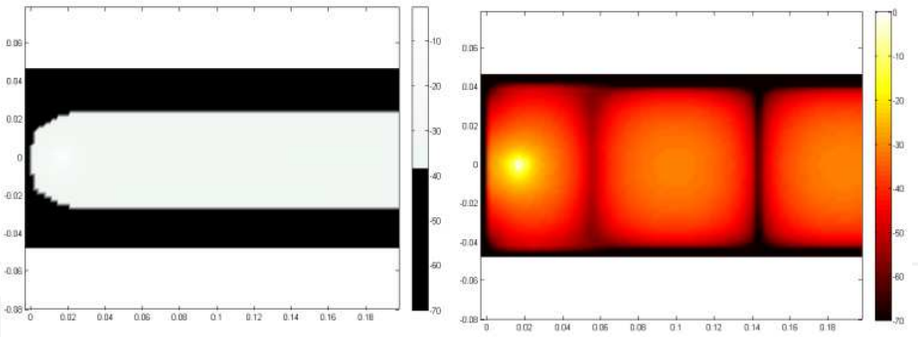


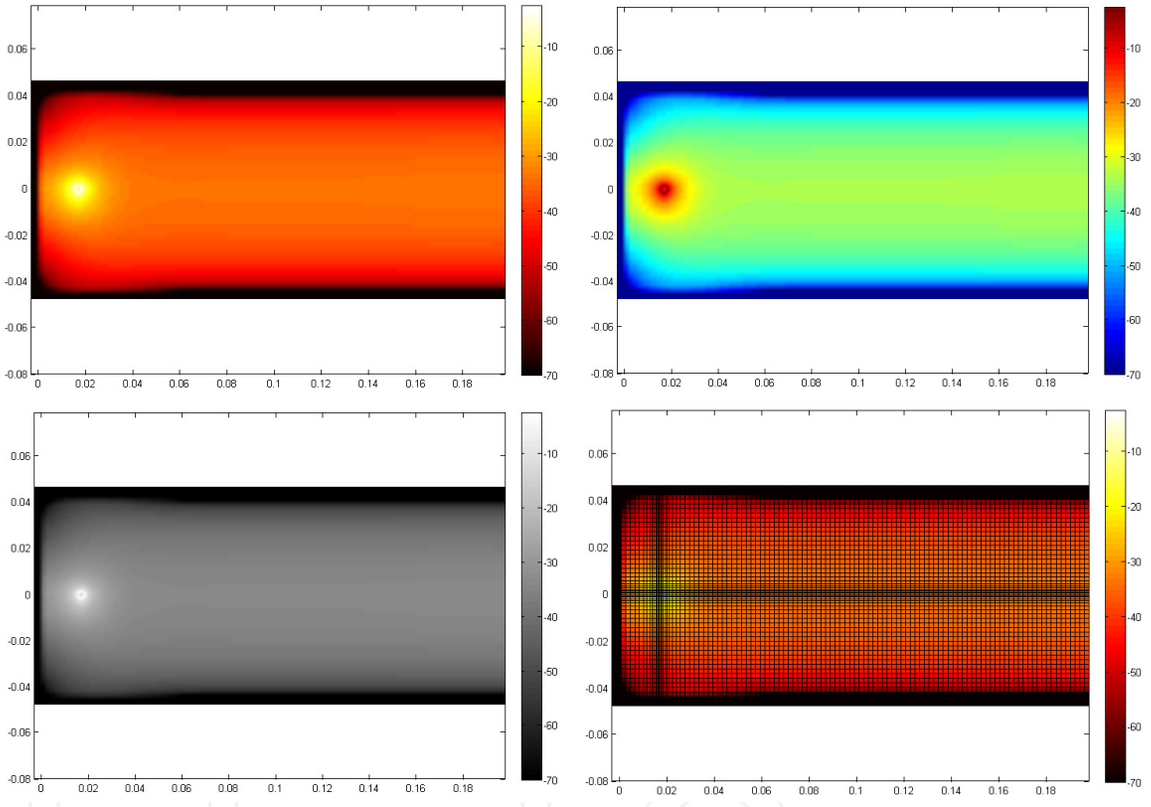
**Figure 3.** Model Setup and Its Voxel Representation (waveguide section with excitation probe at 2.45 GHz, voxels shown in section)

We simulated simple section of a waveguide (inner dimensions 100x50x200 mm) with one side shorted (there is an excitation probe in form of a cylinder in the distance of 17 mm from the shorted end) and the other side open (absorbing boundary condition – absorbs 99.9% of incident power). We extracted the data and plotted them using Matlab.

As you can see (Fig 4ab.) we used several colormaps which enable us to highlight different aspects in our results interpretation. Sometimes it is needed to have contrast colormap (jet, lines – for more information see Product Help of Matlab), at other circumstances you may need to use fine and moderate colormaps (hot, gray, bone, pink – for more information see Product Help of Matlab). In the fourth graph in Fig. 4 we used different shading – faceted. This enables us to highlight structure of computational grid. In many commercial simulators of EM field parts of a model need to be meshed finer than others (e.g. in our case the excitation probe needs to be meshed four times more than the rest of the waveguide to be voxeled sufficiently). In other examples we used shading interp to get more clear view.



a - Interpretation of Extracted Data (Y-plane, middle section of the waveguide) [dB]



b - Interpretation of Extracted Data (Y-plane, middle section of the waveguide) [dB]

**Figure 4.** a - (*RMS*|*E*| colormap-Custom, real modulus E in phase 0° colormap-Hot)
b - (from left upper corner to right lower corner: *RMS*|*E*| colormap-Hot, *RMS*|*E*| colormap-Jet, *RMS*|*E*| colormap-Gray, *RMS*|*E*| shading faceted)

We can also utilize custom colormaps. This can be exceptionally beneficial in applications where we need to find out where values are at some critical level or higher. We illustrated this feature in the first image in Fig. 4a. In Fig. 5. there is an Colormap Editor which can be accessed through: Figure – Edit – Figure Properties – Colormap pull-down menu – Custom.

In our example we set segment in the middle to black colour and segment next to it to white colour. This resulted in the graph as seen in Fig. 4a. For more information on colormaps please refer to the Product Help of Matlab.

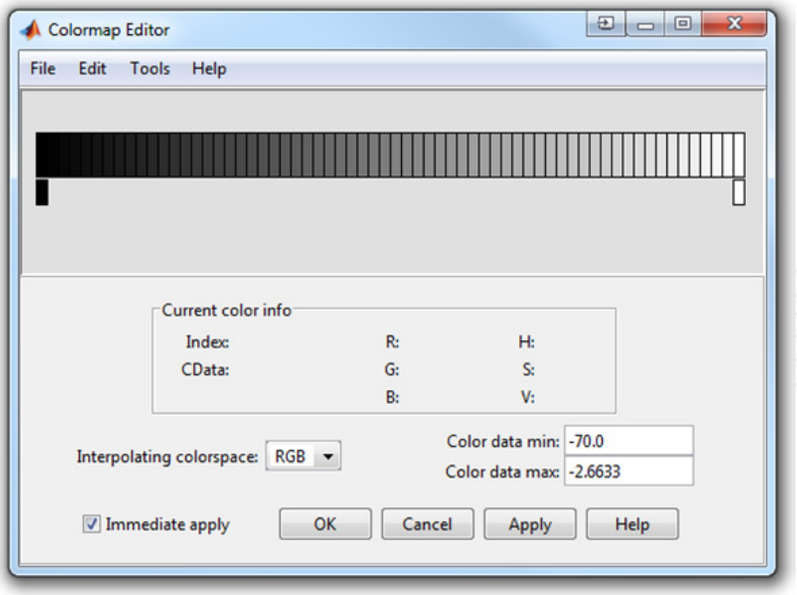



**Figure 5.** Colormap Editor Window

Furthermore, we illustrated how real modulus of vector of intensity of electric field at phase 0° is interpreted using Matlab (second image in Fig. 4a.). This is the most basic interpretation of obtained data we can do.

Note that this kind of results interpretation is much more flexible than the interpretation allowed by post processing tools in commercial EM simulators. In the following example we shall show how to work with time dependency of phasors. Since the results of EM field simulator are extracted when the steady state is reached time dependency is reduced to angle of phasors depicting the field of vectors. Through the following method we can alter phase of those phasors and show real part and imaginary part through one period. The results can be seen in Fig. 6. Figure 7. shows example of data processing to achieve this.

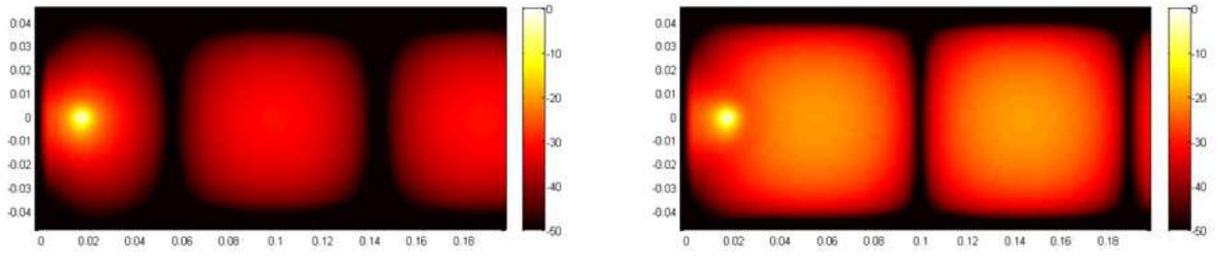


**Figure 6.** Phase Shifted Data – real part of vector ***E*** [dB] (left phase = 0°, right phase = 90°)

Note: In many EM field simulators you may encounter various errors. Pay special attention to the data structure of your exported data since it may not be useful in the way we have shown here (e.g. real and imaginary parts are exported as absolute values so the vital information about phase is lost).

Note: In the part of the script (Fig. 7.) where lowerThan variable is used we are changing the range of values. Since there are parts of model where values of intensity of electric or

magnetic field are very near to zero, minimum of these values in dB would be around -400 dB. This renders produced images useless (value range is huge but most of the relevant values are in the region <-50,0>). Using function find we identify indexes of elements with values lower than -50 dB and we replace those elements with value -50 dB. For more information on function find please refer to the Product Help of Matlab.

Note: In this example we use function subaxis which has similar usage as function subplot but allows users to set the layout of plots in the figure more accurately (options Padding, Spacing, Margin etc.). For more information on subaxis please see internet documentation.

### *3.2.2. Treatment efficiency analysis*

In the following example we will go through one of many useful applications of EM simulators today – evaluation of hyperthermia cancer treatment. Generally, hyperthermia is a method through which tissue is overheated (usually using microwave energy) and cells die (principle of this method is that energy is focused into the cancerous tissue which is less perfused and thus it is more heated – temperature in treated area rises above levels that trigger cell apoptosis). For more information on microwave hyperthermia see for example (Vrba & Oppl., 2008).

Model of simulated experiment can be seen in Figure 8. In this example we use waveguide applicator which is fed by coaxial line ending in protruding inner wire which is located near the shorted end of the waveguide section. There is a horn aperture which helps focus microwave energy to the desired area.

Relatively complex and complicated structure of human body is replaced with so called "phantom" that represents simple muscle tissue. In the model there is a tumor located 1 cm below the surface of phantom. This tumor has the same dielectric parameters as the surrounding muscle tissue (usually the only difference in simulations between muscle tissue and tumorous tissue is in their perfusion, heat transfer rate and heat generation rate – generated heat in tumorous tissue is usually transferred slower than in physiological surrounding tissue).

Additionally there is a water bolus which serves as a coolant body protecting the surface tissue of patient and moving the maximum of temperature to the lower layers. In this example we use source at 434 MHz and the applicator is filled with water (required dimensions of the applicator are effectively lower and impedance matching between waveguide-bolus and body are much better). For more information on waveguide hyperthermia applicators please refer to (Vera et al., 2006).

In this simulation we again extract rough data from EM simulator and we process them further using Matlab. We again have intensity of electric and magnetic field defined in every element of the model. For effective analysis of the treatment we need to evaluate SAR (Specific Absorption Ratio) which describes how much power is absorbed in a weight unit [W/kg]. For more information on SAR see http://www.ets-lindgren.com/pdf/sar_lo.pdf.

```
86
87
88      % In previous process we separated only one slice from the 3D matrices
89 -    complexMatrix = exportedRealPart + 1i.*exportedImaginaryPart;
90 -    phi = [0,pi/2]; % variable phase shift
91 -    cMatrixDimensions = size(complexMatrix);
92
93 -    logMatrix = zeros(cMatrixDimensions(1),cMatrixDimensions(2),length(phi));
94
95 -    for k = (1:length(phi))
96 -        phaseShiftedMatrix = complexMatrix.*exp(1j.*phi(k));
97
98 -        realMatrix = real(phaseShiftedMatrix);
99
100 -       hValue = max(max(abs(realMatrix)));
101
102 -       logMatrix(:,:,k) = 20.*log10(abs(realMatrix)./hValue);
103
104 -       lowerThan = find(logMatrix < -50);
105 -       logMatrix(lowerThan) = -50;
106 -   end;
107
108 -    d=figure(k); clf;
109 -    subaxis(1,2,1,'Padding',0.001,'Spacing',0.001,'Margin',0.03); hold on;
110 -    pcolor(osaY,osaX,logMatrix(:,:,1));
111 -    colormap(hot);
112 -    shading interp;
113 -    axis image;
114 -    colorbar;
115 -    subaxis(1,2,2,'Padding',0.001,'Spacing',0.001,'Margin',0.03); hold on;
116 -    pcolor(osaY,osaX,logMatrix(:,:,2));
117 -    colormap(hot);
118 -    shading interp;
119 -    axis image;
120 -    colorbar;
121
```

**Figure 7.** Method of phase shifting results

To determine how SAR is distributed we need to use this formula.

$$SAR = \frac{\sigma}{2\rho}|E|^2 \tag{4}$$

As we see, in this case we can very well utilize $RMS|\boldsymbol{E}|$. From formula (3) we can say that SAR is defined by following equation.

$$SAR = \frac{\sigma}{\rho} RMS|E|^2 \tag{5}$$

SAR is thus depending on RMS value of intensity of electric field, on conductivity of a material and on its density. We need to obtain those values somehow. In our case whole phantom has homogeneous density and electric conductivity thus we can only mask matrix of $RMS|E|$ and multiply each element by coefficient produced by ratio of σ and ϱ.

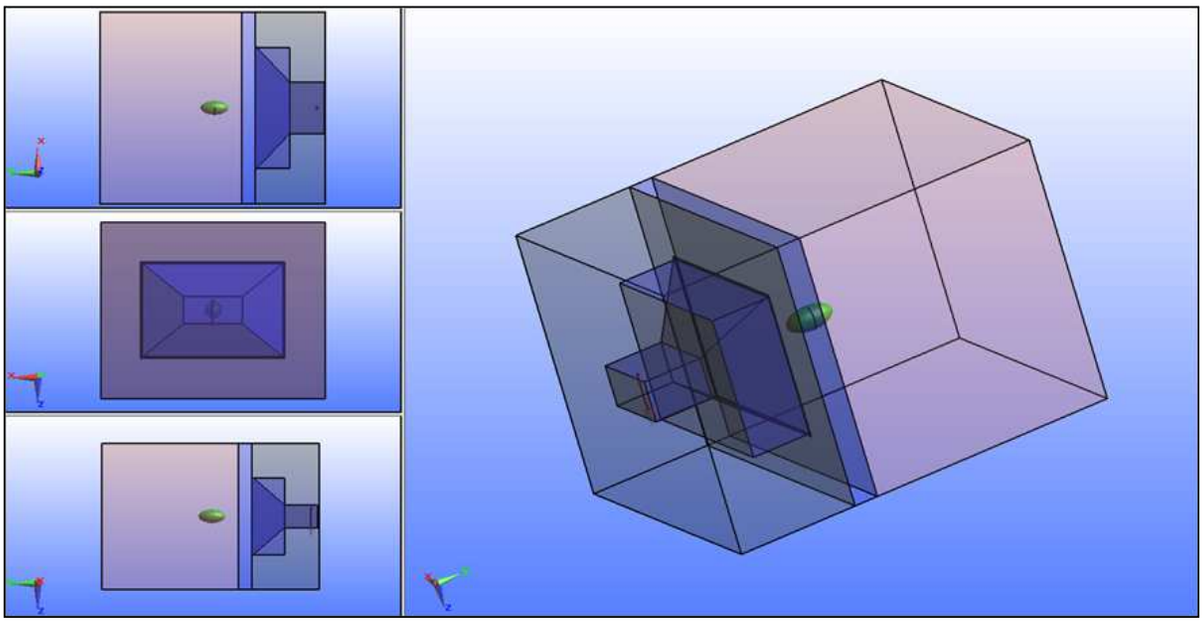


**Figure 8.** Model of microwave hyperthermia at 434 MHz

Note: There are other options how to do that. For example some EM simulator allow users to export matrix of dielectric parameters of a model (this will produce similar matrix to our masking matrix).

Now we should look into how to produce masking matrix efficiently. From our model we know that every element of the voxelized model is phantom if its value of y axis is higher than 120 mm (we can find out this in CAD part of EM simulator). We can prepare our three dimensional matrix of $RMS|E|$ according to previous sections. We also have actual axis of this matrix which we obtained from exported file as well. We simply need to identify which element is at the 120 mm value and then we shall mask the elements with lower value of y axis.

An elegant way to find the index at y axis of element which has the value nearest to 120 mm (note that the value is not usually exactly 120 so if we looked for the exact match we would probably fail) is to use function `min`. Example of this method is as follows.

```
lookFor = 120;
[min_difference, array_position] = min(abs(araay-lookFor));
% Minimal difference shows us the difference between the closest element
% and our looked for value. Array position then indicates the position of
% such element.
```

**Figure 9.** Example of finding the closest element position

In this example, we can illustrate one other useful feature of Matlab processing of EM simulator results. We can instantly normalize obtained results. Usually it is possible to extract actual power in the source which was simulated (e.g. 0.002496 W) and we can utilize this to obtain normalization coefficient. Power normalization coefficient is given by the following formula.

```
if min_difference > 0
     values(:,((position-1):-1:1),:) = 0;
     values(:,(position:length(actualAxisY)),:)...
     = (values(:,(position:length(actualAxisY)),:)).^2.*sigma./rho;
else
    values(:,((position+1):-1:1),:) = 0;
    values(:,(position:length(actualAxisY)),:)...
    = (values(:,(position:length(actualAxisY)),:)).^2.*sigma./rho;
end;
```

**Figure 10.** Covering Field of $RMS|E|$ with Mask

$$coef\ f_{power} = \frac{Power\ to\ which\ we\ want\ to\ normalize}{Actual\ Simulated\ Power} \tag{6}$$

Since this is the coefficient of power ratio we need to use coefficient of intensity of electric field ratio which is the square root of coefficient of power (see the equation below).

$$coef\ f_{power} \approx \frac{E^2_{norm}}{E^2_{acctual}} \Rightarrow coef\ f_E = \sqrt{coef\ f_{power}} \tag{7}$$

Simply by multiplying the matrix of data by this coefficient we obtain normalized data to the desired value of delivered power.

Now we can show several interpretations of results we obtained (Z section in the middle of the model). Thanks to Matlab's flexibility and versatility we can highlight the results in the way that is far more efficient than by using post processing of the EM field simulator. In the following figures we show intensity of electric field and SAR in the treated area.

As you can see in figure 11. each colour scheme highlights different thing (as for the first three). In the fourth graph we used custom colormap to show some kind of a critical zone. In this example we expected that higher values of SAR than 1000 W/kg are considered to be dangerous in some regions (note that these critical limits differ significantly in every application – many factors contribute, e.g. vital or sensitive organs near the treated area etc.) and thus we highlighted the zone with higher values using bright red colour. This may be very important but commercial simulation software generally omits such option.

It may be also very useful to show obtained data only in the region of the tumour or to show values in form of several slices, semi-transparent layers or iso-surface view. Now we shall show how to prepare masking matrix for some simple geometrical objects representing tumours. For more information on the mentioned advanced techniques of results representation please see section 4. Advanced Viewing Techniques.

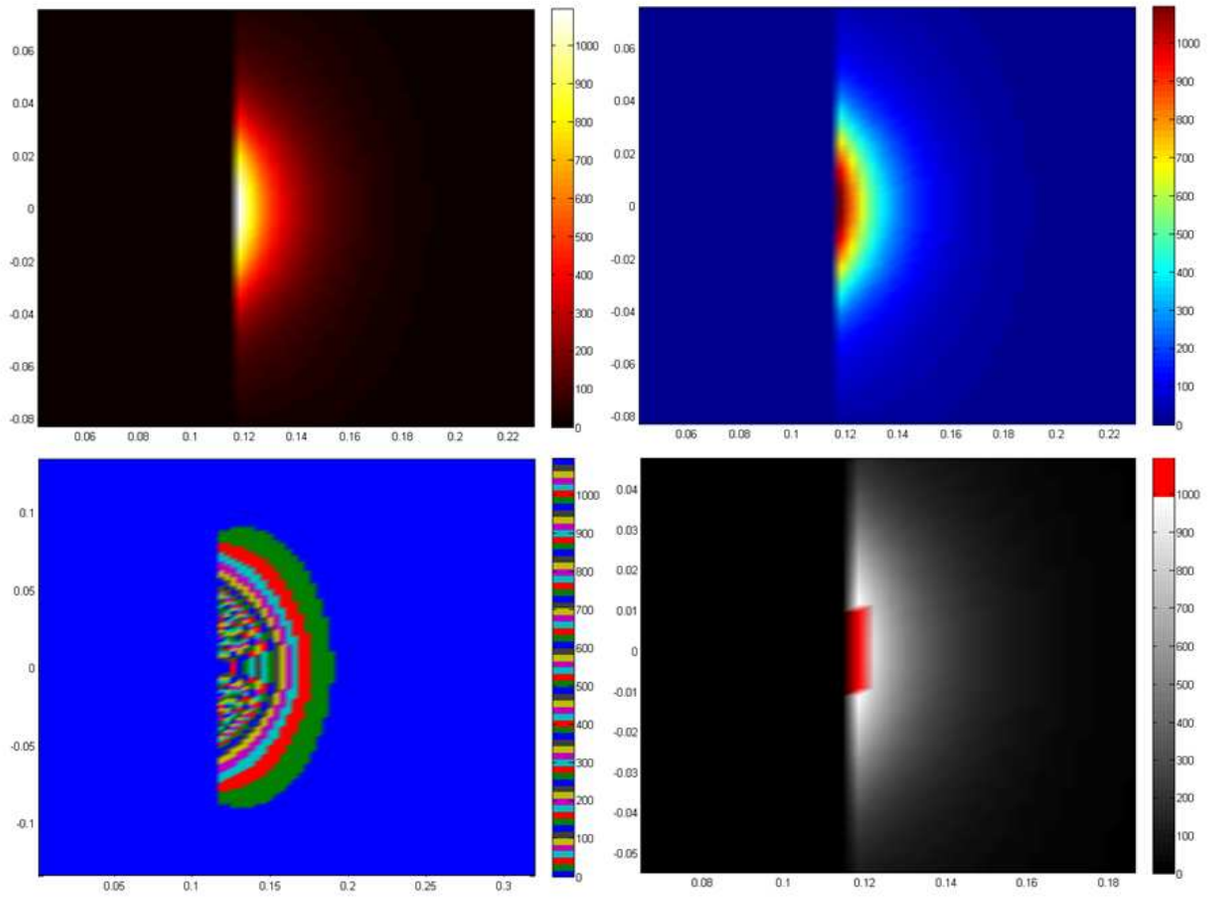


**Figure 11.** SAR [W/kg] in Different Colour Schemes (from left upper corner to right lower corner: hot, jet, lines, custom)

### *3.2.3. Preparation of masking matrix*

Very simple masking (i.e. when the region that needs to be masked is in the form of a cuboid) was shown in previous section. Now we should look into masking regions round in shape (i.e. spheres, ellipsoids and other common shapes that parts of model can be represented by).

Spheres are case of ellipsoid and we shall treat them as such. So our main aim now is to mask region generally in ellipsoidal form. Equation defining an ellipsoid (with its origin at $x_c$, $y_c$, $z_c$) in the three dimensional coordinate system is shown below.

$$\frac{(x-x_c)^2}{a^2}+\frac{(y-y_c)^2}{b^2}+\frac{(z-z_c)^2}{c^2}\leq 1 \tag{8}$$

The graphical representation of a general ellipsoid can be seen in figure 12. There we can see that $a$, $b$ and $c$ are semi-principal axes of ellipsoid and define its dimensions.

Albeit there is a built-in function designed to generate ellipsoid (see the Product Help of Matlab) we shall show you an easy way which allows you to generate desired masking matrix (e.g. with values 0 or 1).

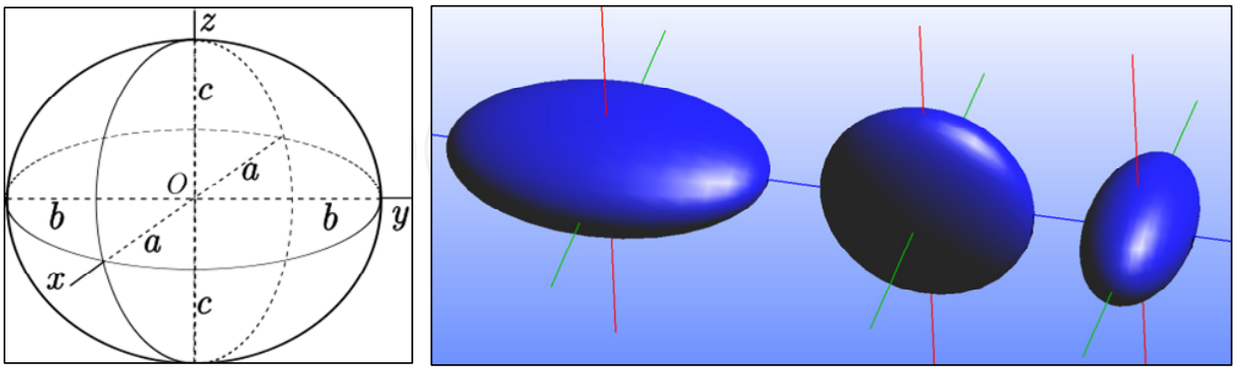



**Figure 12.** Representation of Ellipsoid (from left: scheme, tri-axial, oblate, prolate)

As you can see in Fig. 13. we simply utilized the formula representing ellipsoid and all elements whose coordinates meet the given restrictions are filled with ones. The other elements remain zero. Constants `elementsX`, `elementsY` or `elementsZ` define the length of axis of computational domain (i.e. `actualAxisX` and so on). Variables `xc`, `yc` and `zc` define centre of our ellipsoid.

```
85 -        maskingMatrix = zeros(elementsX,elementsY,elementsZ);
86          % Note that the size of a maskingMatrix will be the same as the size of
87          % matrix containing the results. We can use this to our advantage and
88          % define maskingMatrix using size[resultsMatrix].
89 -        for x = (1:elementsX)
90 -            for y = (1:elementsY)
91 -                for z = (1:elementsZ)
92 -                    if (((actualAxisX(x)-xc)^2/a^2)+...
93                              ((actualAxisY(y)-yc)^2/b^2)+...
94                              ((actualAxisZ(z)-zc)^2/c^2)) <= 1;
95 -                      maskingMatrix(x,y,z) = 1;
96 -                   end;
97 -               end;
98 -           end;
99 -       end;
```

**Figure 13.** Defining Ellipsoid in the Masking Matrix

Now we can simply use generated masking matrix and multiply every element of result matrix by according element of masking matrix. Then we can look how our results are interpreted in detailed view in the tumour.

From masked matrix of SAR we can also easily extract total power radiated to the tumorous region. Total power lost in the healthy tissue is than given by the difference between total power (in the case of perfectly matched source) and power lost in the tumour.

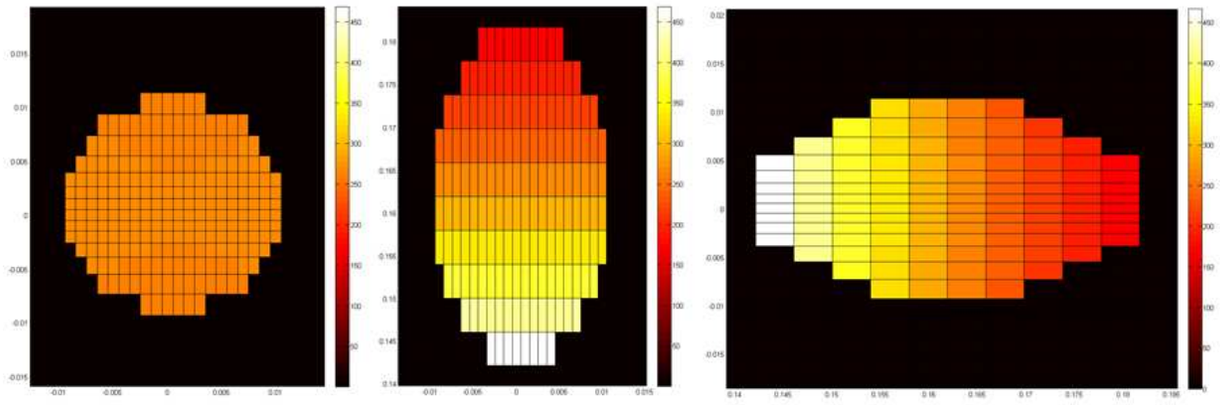


**Figure 14.** SAR [W/kg] in Tumour (from left: Y-slice, X-slice, Z-slice)

Note: In some cases it might be beneficial to set the colorbar range the same in all pictures (as shown in Fig. 14.). The simplest way to do this is to set value of [1,1] element of displayed matrix (i.e. a slice) to the maximum value of the whole three dimensional matrix (if needed, the minimum value should be set to element [max,max] of displayed matrix).

Additional information that we can obtain from masked result matrix is total absorbed power in the tumour. Since we know the dimensions of the tumour we can easily calculate its volume and use following formula.

$$P_{absorbed} = Volume \sum_{all\ voxels} RMS\left|E\right|^2 \sigma \tag{9}$$

The volume of standard ellipsoid is given as follows.

$$V = \frac{3}{4}\pi abc \tag{10}$$

For example in our case tumour has volume only 8.37766e-6 $m^3$. Information on total absorbed power may be very efficient way of preliminary evaluation for the treatment planning. Through this method we can determine power lost in any part of a simulated model. There may be some intricate volumes which are not as easily described as ellipsoid. Then we need to export their volume or masking matrix from the EM simulator (if possible).

We can determine the volume of such complicated shapes by summing elementary volumes of voxels representing those shapes. This may be unnecessary in some more advanced EM field simulators since they allow us to export model data in many suitable forms for Matlab processing. But the method presented in the following text is universal and can serve for better understanding of the model, its grid and working principals of EM simulations.

First of all we need to define which voxels are occupied by the model we want to evaluate. For this example we shall use our previously generated ellipsoid. As you can see in Fig. 14. generally grid of a simulation does not have to be symmetrical (i.e. voxels are not cubes but they are in the form of general cuboids). This means that each element may be representing

different volume. We have our matrix of zeros and ones which we generated in Fig. 13. Now we need to apply this matrix to the actual coordinate system.

We can use built-in function `meshgrid` to produce three dimensional matrices which contain coordinates for specified element (for more information on function `meshgrid` please refer to the Product Help of Matlab). Now we can easily determine exact coordinates of elements occupied by our model (see Fig. 15.).

```
 7 -    oldValue = 1;
 8 -    for k = 1:length(actualAxisZ)
 9 -        [D,E] = find(valuesMasked(:,:,k) > 0);
10 -        if isempty(D) == 0
11 -            C(oldValue:(oldValue + length(D) - 1)) = k;
12 -            oldValue = oldValue + length(D);
13 -        end;
14 -        if k == 1
15 -            A = D';
16 -            B = E';
17 -        else
18 -            A = [A D'];
19 -            B = [B E'];
20 -        end;
21 -    end;
22
23 -    completeVolume = abs(actualAxisX(A + 1) - actualAxisX(A)).*...
24          abs(actualAxisY(B + 1) - actualAxisY(B)).*...
25          abs(actualAxisZ(C + 1) - actualAxisZ(C));
26
27 -    sum(completeVolume)
```

**Figure 15.** Determination of Coordinates of Model Elements

At this moment we need to get better understanding of how models are defined. Since the values (in this case ones) are located in the nodes of the grid not in the centers we need to determine how the model actually looks like. See the following figure for better understanding (2D-plane is used to illustrate).

As you can see there will be an error caused by this node to cell transformation. When the simulation grid is defined appropriately (i.e. it is fine enough) the error will be only marginal. If such error is unacceptable more advanced techniques of node to cell transformation should be used. But for purposes of this example this method is more than sufficient. For example, volume of previously defined ellipsoid determined voxel by voxel is 8.5302e-6 $m^3$. This means that the relative error is 1.8 % (and this error includes error caused by voxelization itself – this means that node to cell transformation really brings only marginal error in simulations with fine grid).

In the Fig. 16. you can see that node to cell transformation can be done in a few (precisely 8) ways. We can easily determine which way is the most precise (i.e. [(A+1) – A] or [(A-1) – A] and its combinations with B and C). Since our model is voxelized symmetrically there is

almost no difference between volumes computed accordingly to various node to cell transformations (ranging between 8.5302e-6 and 8.5304e-6).

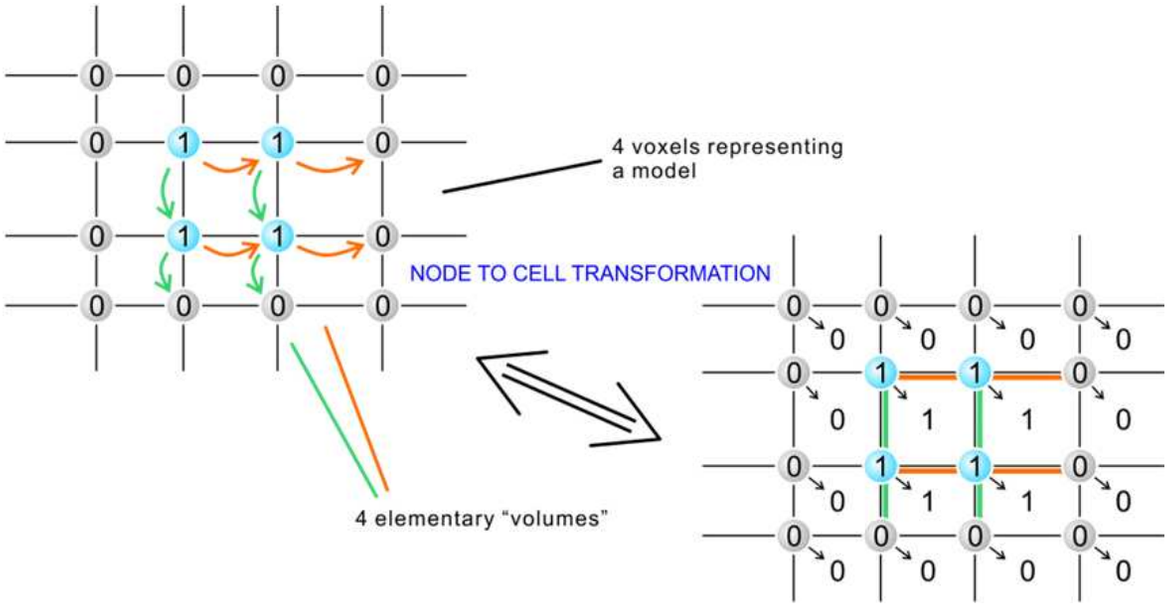



**Figure 16.** Actual Model Dimensions

## 4. Advanced viewing techniques

This section deals with some examples of advanced viewing of results that we have found useful during our work. As you have noticed, up to this point we have represented 3D results only in 2D graphs. This is because usually these 2D representations are clearer and easier to interpret. But for overviewing or demonstrational purposes we might need to show whole situation in 3D. For doing this we shall use multiple sections of a computational domain (`slice` or `surf`) and iso-surfaces.

### 4.1. Multiple layer viewing

We can display results in multiple 2D sections with variable transparency. This method of results visualization can give us much better overview of how a situation looks like.

It can be also beneficial to use surface view as semi-transparent layers to view results in semi-masked way. Through this we can highlight some regions of model without completely blocking visual output from other regions. For example, in our case, we can use one surface view of $RMS|E|$ in agar phantom and semi-mask it with alphamap of tumour (values 1) and the rest of tissue (values 0.5).

Figure 19. shows us how this graph is plotted and how `transparencyMatrix` is prepared.

Note that when using this type of transparency mapping maximum and minimum values must be present in `transparencyMatrix` (i.e. one edge of transparencyMatrix is set to 0 to denote the minimum value of transparency).

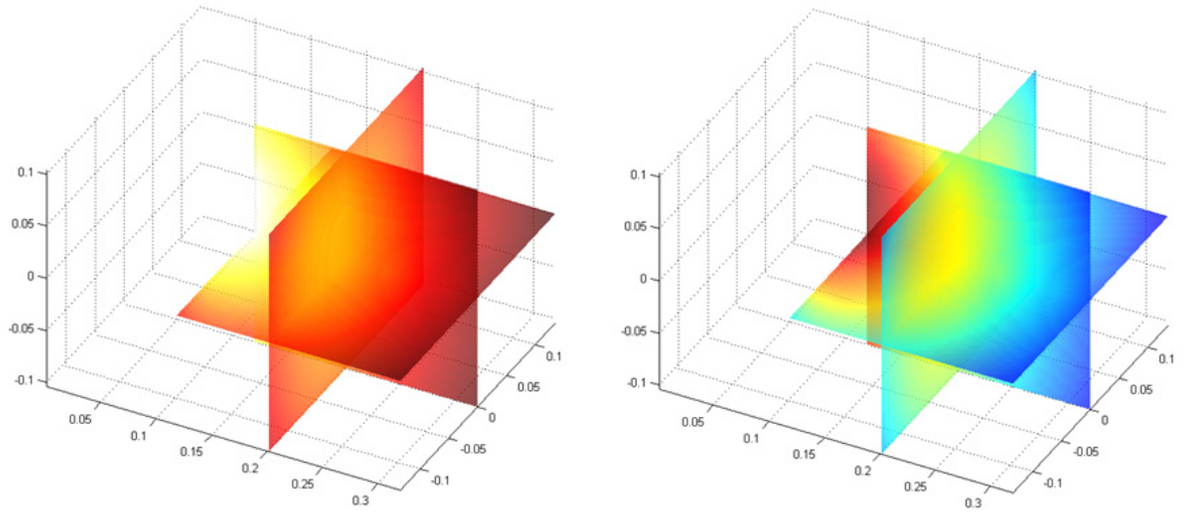


**Figure 17.** Sliced View of $RMS|E|$ [dB] (only in agar phantom, colormap hot, jet, alpha 0.7)

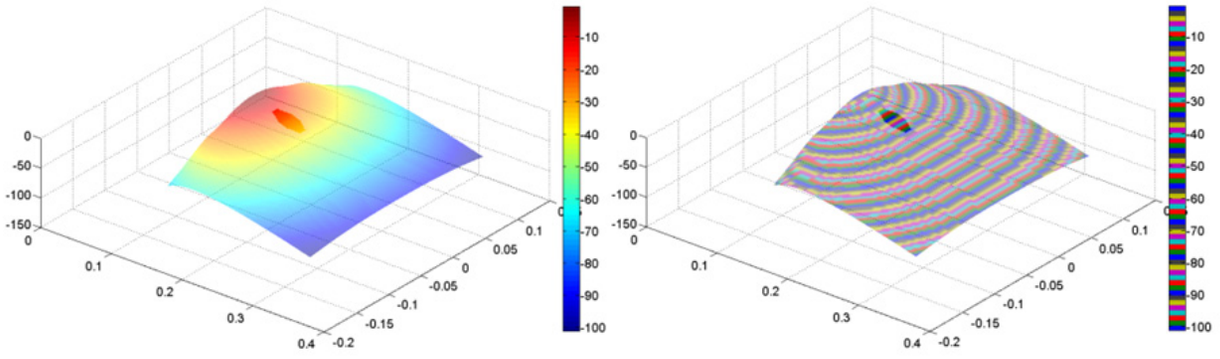


**Figure 18.** Surface View of RMS|E| in [dB] (transparency of tumour region set to 1 – opaque, transparency of other regions set to 0.5 – semi-transparent; colormap – jet and lines)

Through this method we can produce much more intricate visualizations of results which are unseen in commercial EM simulators. For example we can mask results in uninteresting regions (immediate vicinity of power source etc.) almost entirely, semi-mask results in exposed tissue and left results in tumour unmasked, vital organs and other key regions of simulation domain. Note that masking results with transparency matrix does not alter presented values.

This may pose a problem when values around a power source are extreme and render the rest of results unclear. For this purpose you may consider utilizing simmilar approach as presented in Figure 7. Instead of finding values lower than some value you may find values higher than some reasonable value and lower all higher values to this level.

Many more possibilities are offered thanks to post processing of EM results in Matlab. As mentioned before, the greatest advantage is that it is extremely flexible and can meet very specific requirements which can arrise with various applications of EM field. In the following section we shall demonstrate some results obtained during our primary research of EM field of microtubules (nanostructures in living cells which serve as a crude frame of a cell and have other important roles in life cycle of a cell).

```
104 -       transparencyMatrix = zeros(elementsX,elementsY,elementsZ);
105         % Note that the size of a transparencyMatrix will be the same as the
106         % size of matrix containing the results. We can use this to our
107         % advantage and define maskingMatrix using size[resultsMatrix].
108 -       for x = (1:elementsX)
109 -           for y = (1:elementsY)
110 -               for z = (1:elementsZ)
111 -                   if (((actualAxisX(x)-xc)^2/a^2)+...
112                             ((actualAxisY(y)-yc)^2/b^2)+...
113                             ((actualAxisZ(z)-zc)^2/c^2)) <= 1;
114 -                       transparencyMatrix(x,y,z) = 1;
115 -                   else
116 -                       transparencyMatrix(x,y,z) = 0.5;
117 -                   end;
118 -               end;
119 -           end;
120 -       end;
121
122 -       transparencyMatrix(length(actualAxisX),:,:) = 0;
123
124 -       figure(1);clf;
125 -       surf(actualAxisY(40:end),actualAxisX,valuesLog(:,40:end,51),...
126             'FaceAlpha','flat','AlphaDataMapping','scaled',...
127             'AlphaData',transparencyMatrix(:,40:end,51));
128 -       colorbar;
129 -       shading interp;
130 -       set(gca,'FontSize',14)
```

**Figure 19.** Fig. 19. Generation of Overalaying Transparency Matrix and Its Utilization

## 4.2. Iso-surface viewing

In this section we are going to show several results of EM field around microtubule. This structure is generally consisting of protofilaments which are polymerized tubulin heterodimers. Thirteen protofilaments bound together form a microtubule structure which resembles a long hollow tube (see Fig. 20).

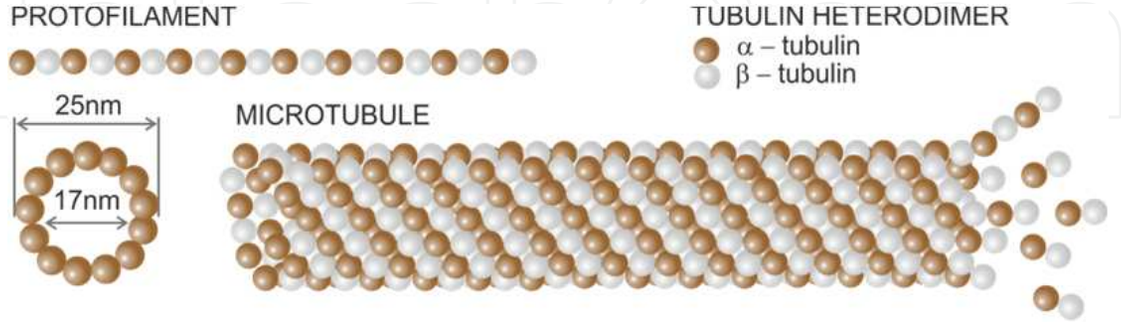



**Figure 20.** Microtubule Structure Visualization

Tubulin heterodimers (i.e. basic building block of microtubules) are highly polar structures and provided some form of external energy (movement of microtubules, mechanical vibrations etc.) they may produce EM field around themselves (Pohl, 1981; Fröhlich, 1978).

In our work EM field simulations of microtubule model were entirely conducted using Matlab. Tubulin heterodimers were represented as vibrating elementary electrical dipoles (EED) and EM field was determined for each of these EEDs. Combining the results led to unravelling the EM field produced by these complicated structures at whole (Havelka, 2009).

For the purpose of this text we shall look at visualization of these results we used to present obtained EM field. Because microtubules are symmetrical structures we have found out that representing its field by iso-surface view is very clear and easy to interpret.

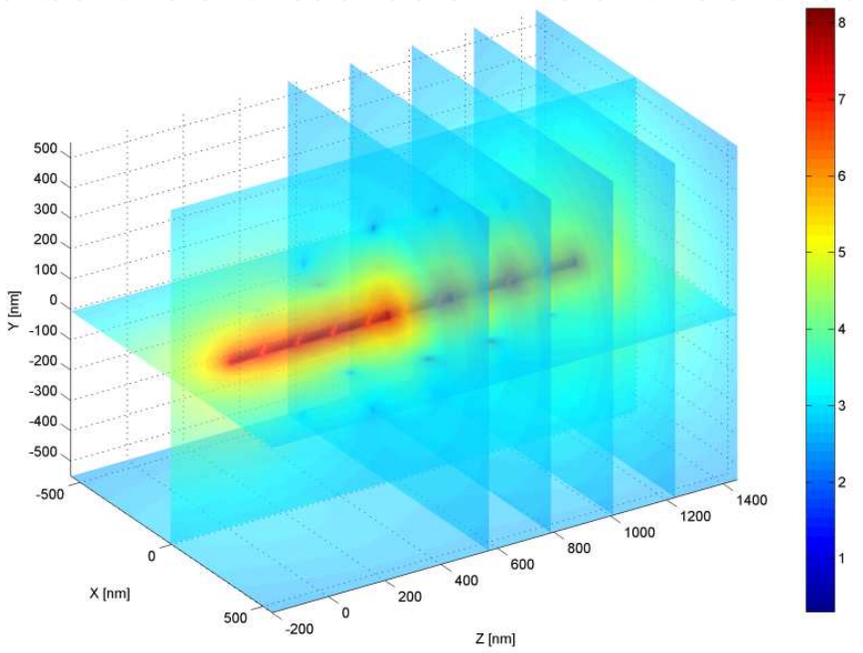



**Figure 21.** Sliced Semi-transparent View of Electric Field around Microtubule (see 4.1)

In the figure 23. you can see part of the code used to generate such visualization. More information on isosurface can be found in the Product Help of Matlab. In this particular example we wanted to view results in the form of several iso-surfaces. First of all we determined the range of data obtained through Matlab analysis of EM field around our sample microtubule.

Then we need to choose which values we want to be visualized as iso-surfaces (in this case, values range is extremely wide therefore we choose only exponents – 7, 5, 2). Then we need to find which positions at colour scale are lower than our actually viewed value (we generate the colour scale using function `jet(length of colour scale)`). Then we simply use function `patch` (to build 3D wire model of locations with desired values – i.e. 1e2, 1e5, 1e7) and we set its colour which we choose accordingly from our generated colour scheme.

Additionally you can choose lighting (particularly useful in this case is `gouraud` lighting which does not produce any glances on multiple iso-surfaces). It is also beneficial to use alpha which is lowered with each loop (to retain clarity of the visualization).

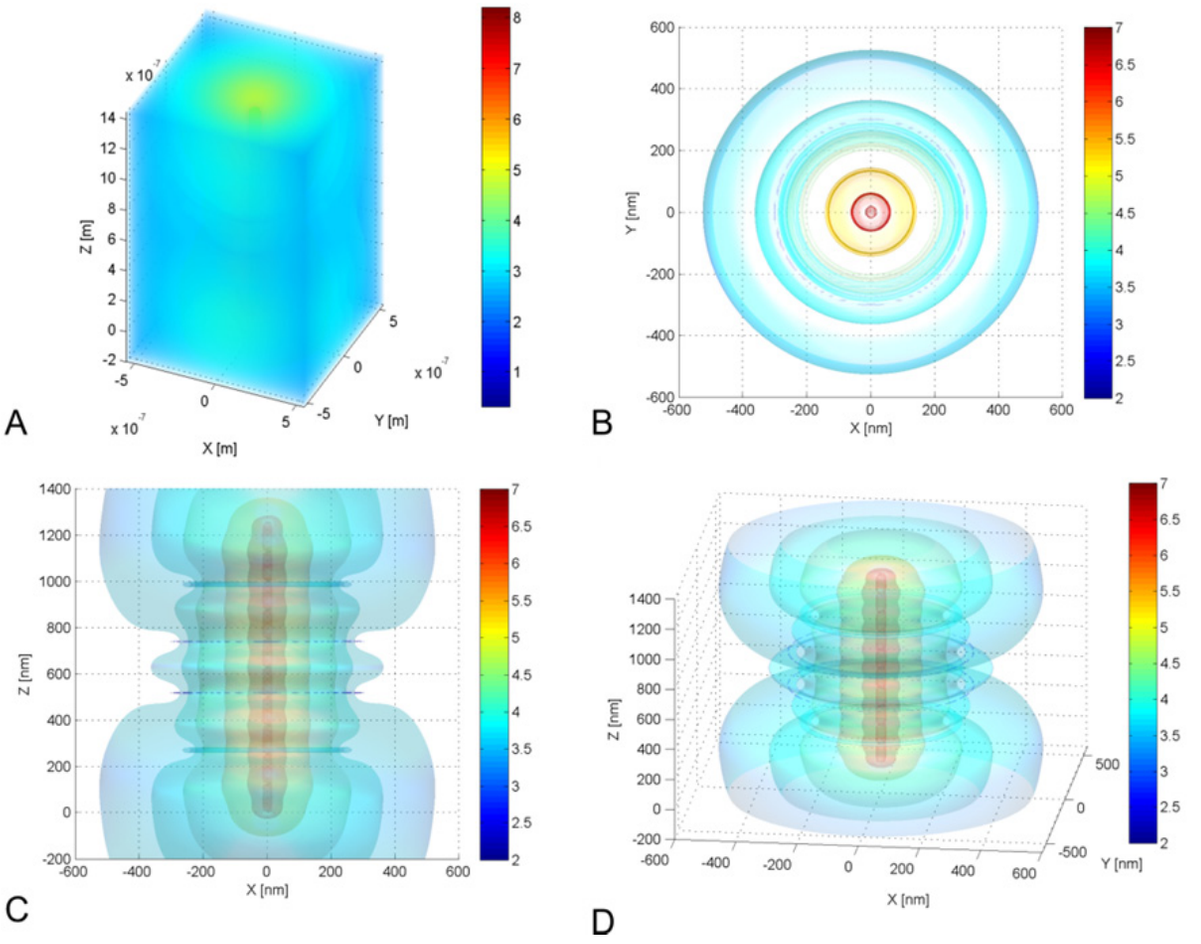



**Figure 22.** Iso-surface View of Electric Field around Microtubule [-] (A – rough data, B – iso-surface view from above, C – iso-surface view from enface, D – iso-surface view – general angle)

Last part is generating tiny triangle which is concealed in the middle of viewed data and contains the minimum and the maximum values which allows us to show colorbar appropriately for all the iso-surfaces .

## 5. Conclusion

In this work we have presented basic methods of how to process obtained rough data from commercial EM simulators or even process data from our own simulations done in Matlab. We show how Matlab processing can be greatly beneficial in highlighting results in many ways that are unseen in commercial EM simulators.

We present very simple way to modify data in the form suitable for further processing and then we illustrate how to view these data in ways highlighting specific aspects (i.e. values in specific regions (tumorous tissue), evaluation of treatment efficiency, utilization of Matlab in primary research, 3D viewing etc.). These innovative ways of combination of specialized software with researcher's versatile tool, such as Matlab is, yield in very productive and efficient way of scientific exploration of the vast field of electromagnetism.

```
max_E = log10(max(max(max(E_matrix)))); % Range of values in imported data
min_E = log10(min(min(min(E_matrix)))); % Range of values in imported data
ISOsurfaces = [7 5 2]; % Which values should be visualized
color_scale = 1000; % Refinement of colorbar
for u = 1:size(ISOsurfaces,2)

  T_vp = find(ISOsurfaces(u) >=...
         linspace(min(ISOsurfaces),max(ISOsurfaces),color_scale));
  color_all = jet(color_scale);

  figure(3);hold on
    p = patch(isosurface...
        (Xcoord*1e9,Ycoord*1e9,Zcoord*1e9,E_matrix,10^ISOsurfaces(u)));
    set(p,'FaceColor',color_all(T_vp(1,end),:),'EdgeColor','none');

    title({'Intensity of electric filed around model of microtubule';...
           'log_{10}(V/m)'});
    xlabel('X [nm]'); ylabel('Y [nm]'); zlabel('Z [nm]');
    grid on;
    camlight; lighting gouraud
    alpha(0.5 - 0.05*u)
    hold on;
    a = [1,1;0,0]*1e-9; b = [0,1;1,0]*1e-9; c = [0,0;0,0]*1e-9;
    d = [max(ISOsurfaces) max(ISOsurfaces) ;...
         min(ISOsurfaces) min(ISOsurfaces)];
    surf(a,b,c,d)
    colorbar;
end;
```

**Figure 23.** Generation of Iso-surface View of EM Field around Microtubule

The results that are obtained in our research of EM field around living cell help us understand crucial facts about this part of our lives which has to be truly discovered yet. Matlab in this instance allows us to visualize result so we can support or on the other hand disapprove many theories (e.g. transportation of particles around microtubule via EM field).

## Author details

Tomas Vydra and Daniel Havelka
*Czech Technical University in Prague, FEE, Department of Electromagnetic Field, Czech Republic*

## Acknowledgement

This research is supported by the Grant Agency of the Czech Republic by project 102/08/H081: "Non Standard Applications of Physical Fields" and by CTU project SGS12/071/OHK3/1T/13: "Advanced techniques in the processing of industrial materials and biopolymers using electromagnetic field - multi-frequency processing ". Further it is supported also by research program MSM6840770012: "Transdisciplinary Research in the Area of Biomedical Engineering II" of the CTU in Prague, sponsored by the Ministry of Education, Youth and Sports of the Czech Republic.

## 6. References

Cifra, M.; Pokorný, J.; Havelka, D. & Kučera, O. (2010). Electric field generated by axial longitudinal vibration modes of microtubule, *BioSystems*, ISSN 0303-2647, vol.100, no.2 (May 2010), pp. 122-131, Oxford, GB

Fröhlich, H.; (1978). Coherent electric vibrations in biological systems and the cancer problem, *IEEE Transactions on Microwave Theory and Techniques*, Vol.26, No.8 (August 1978), pp. 613-618, ISSN 0018-9480

Havelka, D.; Cifra, M. (2009). Calculation of the Electromagnetic Field Around a Microtubule. *Acta Polytechnica*, Vol.49, No.2 (2009), pp. 58-63. ISSN 1210-2709.

Havelka, D.; Cifra, M.; Kučera, O.; Pokorný, J. & Vrba, J. (2011). High-frequency electric field and radiation characteristics of cellular microtubule network, *Journal of Theoretical Biology*, ISSN 0022-5193, vol. 286, no. 1, p. 31-40, London, GB

Pohl, H.A. (1981). Electrical oscillation and contact inhibition of reproduction in cells, Journal of Biological Physics, Vol.9, No.4 (1981), pp. 191-200, ISSN 0092-0606

Thomas, V.A.; Ling, K.; Jones, M.E.; Toland, B.; Lin, J. & Itoh, T. (1994). FDTF analysis of an active antenna, *IEEE Microwave Guided Wave Lett.,* Vol.4, No.9 (September 1994), pp. 296-298, ISSN 1051-8207

Vera, A.; Valentino, A.; Baez, A.; Trujillo, C.J.; Zepeda, H. & Leija, L. (2006). Rectangular Waveguide for Electromagnetic Oncology hyperthermia: Design, Construction and Characterization, *Proceedings of 3rd International Conference on Electrical and Electronics Engineering 2006*, pp. 1-4, ISBN 1-4244-0402-9, Veracruz, Mexico, September 6-8, 2006

Vrba, J.; Oppl, L.; & Vrba, D. (2008). Microwave Medical Imaging and Diagnostics, *Proceedings of EuMC 2008 38th European Microwave Conference*, pp. 408-411, ISBN 978-2-87487-006-4, Amsterdam, Netherlands, October 28-31, 2008

Vrba, J. & Oppl, L. (2008). Prospective Applications of Microwaves in Medicine, *Proceedings of COMITE 2008 14th Conference on Microwave Techniques*, pp. 1-4, ISBN 978-1-4244-2138-1, Prague, Czech Republic, April 23-24, 2008

Vydra, T.; Vorlicek, J. & Vrba, J. (2011). Measurement and analysis of microwave processing of biopolymers, *Proceedings of 21th International Conference Radioelektronika 2011*, pp. 1-4, ISBN 978-1-61284-325-4, Brno, Czech Republic, April 19-20, 2011



IntechOpen