# Necessary Information About the HTP-Project

Erik Bertolino, Erika Ek, Rebecca Jonasson,
Andrea Nygren and Julia Ravanis

July 27, 2017

## 1 Introduction

This document is meant to provide an overview over the state of the project regarding the creation of a Hyperthermia treatment plan. Initially, the current state of the project is presented which entail information about where to find the code, the optimization method, how to select parameters, current issues and a guide over what can be found on Newhopa that is related to this project. Thereafter follows suggestions of future developments to this project.

## 2 Current Situation

The current state of the optimization code can be found in the git-project *Hyperthermia_Treatment_Plan* and to gain personal access it is necessary to contact Julia Ravanis at julia.ravanis@gmail.com. The project is also available on Newhopa in the user-folder *HTP_Project*. It consists of a folder system of which there is an overview in the folder *Information*.

Now, the optimization is performed with particle swarm since this was found to be faster than fminsearch, especially when optimizing over all antennas. Optimizing over all antennas have consistently provided significantly better results. However this could be explained by there seem to be some sort of error when excluding certain antennas from the process.

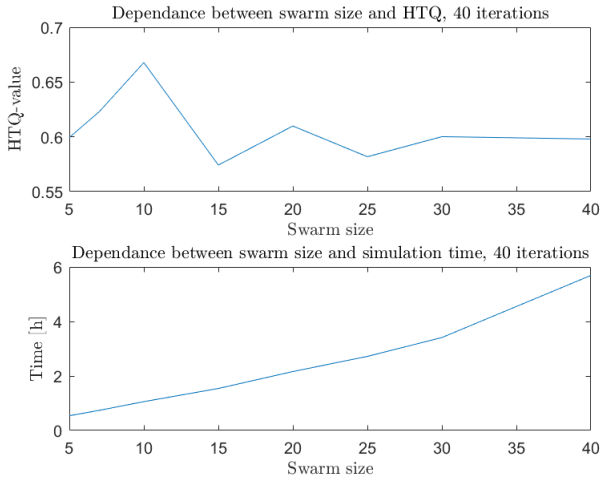
Optimizing over one frequency and combining frequencies two and two seem to work fine as it is now. They consistently provide solutions that are as good as, or better, than focusing in the tumor using settings calculated in Matlab or using time reversal simulations. Results gathered using the current state of the code is provided in Appendix.

## 2.1 Specific Parameters Being Used

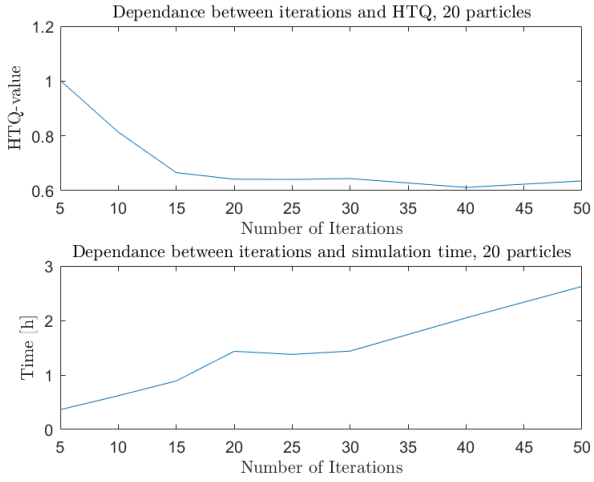
As it is now, most operations are performed with efields converted into an octree-format. This is done using a relative epsilon of 0.1. It has been verified that this generates an insignificant error in PLD distributions compared to the matrix format. Currently the swarm is initialized as a matrix of size $s \times 2N - 1$, where $s$ is the swarm size and $N$ the number of antennas. For each antenna 2 values are initiated that correspond to the real and complex part of its setting. One particle is set to be 1 and 1 for each antenna so that the optimization does not start at a position which is worse than the starting point where all antennas radiate with the same amplitude and phase shift. The randomness of the initialization is necessary for the particles to locate a minimum, however this leads to a comparability issue and a difficulty to generate the exact same result given the same input settings.

The optimization preforms differently depending on number of particles used (swarm size), number of iterations and number of stall-iterations all which are input parameters. Naturally, a large swarm size and a high number of iterations generate a more reliable result but both parameters also affect the simulation time a great deal. Chen, Montgomery, and Bolufé-Röhler (2015) describe how the number of particles are significant only up to a certain limit, in our case under 10 particles seem to be an unstable choice, and above that the number of iterations required for convergence remain stable. In their paper they recommend around 700 iterations for our problem type, however over 50 iterations little improvement in HTQ has been found. Different number of stall-iterations have not been much investigated, it has mostly been set to 10 since the optimization has been known to sometimes stall about 7 times in the beginning.

The graphs in Figure 1 illustrate variation HTQ and simulation time over swarm size and number of iterations, respectively. These are meant to provide an overview of the dependence but this connection might need to be investigated further to provide a guide over which combination that provide the best HTQ in the least amount of time.




(a)




(b)

Figure 1: Subfigure 1a illustrate variation in HTQ and simulation time depending on swarm size nad Subfigure 1b illustrate the same variation depending on the number of iterations.

## 2.2 Existing Issues

As mentioned above, there is a comparability issue of results due to the randomness of particle swarm. This factor also makes it difficult to gain the same

results using the same parameters. To assure convergence and reliability of the results a high amount of iterations are needed and even then it is possible that the swarm find a local minimum in which it gets stuck. Increasing the amount of iterations greatly is currently not an option since the code is very time consuming as it is.

The code being very time consuming is possibly the greatest issue since this makes the prospect of investigating all combinations of many different frequencies next to impossible since it would probably take weeks. The three different optimization selections: *M1-M1*, *M1-HTQ*, and *M2* each take different time to run, *M1-M1* being the fastest and *M2* being by far the slowest. If *M2* is not made more effective it will not be worth the wait to use this option. There is also some cell error in M2 that is probably easy to fix but due to lack of time and it taking so long to run, we were not able to locate the error. One way to lower the simulation time is to remove E-fields for antennas that seem to not contribute much. As it is now, there exists code for this, a function called *select_best*, but there is something that is not working as it should.

The CST-macro for elliptical placement of antennas of different sizes is currently not ideally adjusted for elliptical parameters. It works fine for circular parameters, and for elliptical parameters but when all antennas are of the same scale, but for a placement with both differently scaled antennas and elliptical parameters the spacing between antennas is not evenly distributed. The rotation of the antennas are derived according to a circular placement and there is a calibration which is done to compensate for using elliptical parameters that seem to be effective for placing antennas of the same size. When using differently scaled antennas this calibration is ineffective and, due to lack of time, the antennas are in this case placed according to the angles calculated according to a circle.

## 2.3 Available on NewHopa

In the folder *HTP_Project* there are the following subfolders:

- CylinderModels
- DukeModels
- Hypethermia_Treatment_Plan
- Results
- tissue_files
- tissue_matrices

The optimization git project is updated and stored in *Hypethermia_Treatment_Plan*. Tissue files and tissue matrices of the existing models are stored in the folders *tissue_files* and *tissue_matrices*, respectively. The folders *CylinderModels*, *Child-Models* and *DukeModels* contain the CST models of which there are results in the *Results*-folder. Within the *Results*-folder there are sub-folders corresponding to each model which in their turn contain folders for each 25th frequency from 400-900MHz and folders for time reversal-results. These are antenna settings which are exported from the models that end with *_FFT* and can be used as a comparison for the settings derived from the optimization. In order for the optimization to be considered functional it should provide settings which generate a lower HTQ than the time reversal settings.

There exist exported efields for all frequencies for all models except for the child model. Time reversal results are not exported for Duke-skullBaseL, Duke-nasalWerT and Child-7s7b, however this can be done using the script *RunFFT-calculation* under *D:\MATLAB\FFT Phase and Amplitude from dipole*.

Another way of confirming the optimization results is to derive settings in matlab instead. This can be done using the script *CalculatePhaseAndAmplitude* which is found under *D:\MATLAB\Phase and Amplitude*. This is faster than running a CST FFT simulation and can even provide better results. Here you enter the point of which you want to focus, for instance the center of the tumor, the tissue matrix, and antenna coordinates. The antenna coordinates can be derived from CST by using *pick circle center* for the bottom of the bolus for each antenna. From the pick list it is possible to copy-paste the coordinates into Matlab ant then save them as a $3\times N$ matrix, $N$ being the number of antennas. Don't forget to enter into the code that these are in CST's coordinate system.

Note that the child-model has different antenna sizes that have different frequency spans. This means that efields for different frequencies are to be combined. The spans of each antenna can be observed through the S11-parameter in CST. Antenna placement for the child-model is done by the creation of a transformation matrix in Matlab, in *D:\MATLAB\Antenna placement\Head* on Newhopa. This is then implemented using the macro *Antenna Placement from Transformation File* in CST found under *Home* $\Rightarrow$ *Macros* $\Rightarrow$ *User Defined Macros*. This placement method is not ideal and it is common that manual adjustment of antennas is required.

Two CST-macros are created to handle differently scaled antennas one which is elliptical and one for a sphere. These can be found under *Home* $\Rightarrow$ *Macros* $\Rightarrow$ *User Defined Macros* and are called *Antenna Placement Ellipse For Multiple Scales* and *Antenna Placement Sphere For Multiple Scales*. A closer description of these can be found when they are opened in the macro editor.

# 3 Future Projects

There are several thing to continue working on in this project and it is also necessary to explore alternative methods. One natural way of starting to carry on with this project is of course to address the issues being mentioned in Section 2.2. Investigating the HTQ vs time dependence of combinations of swarm size and number of iterations might provide an idea for the most time effective choice, which could be useful for further simulations. It could also be of interest to study social adjustment and different limits for particle swarm. The optimization needs to be adjusted for using antenna setups with differently scaled antennas.

## 3.1 Multiple frequencies

The optimization should in the future work for more than two frequency combinations. Currently, a script called *combine_single* combines already optimized E-fields and finds a combined HTQ and time shares. This can be used to combine two or more frequencies, but it is only an approximation of how the combination would be. To properly combine different frequencies, the different E-fields need to be combined based on each other. There are two ways to go from here:

1. Continue to develop single and double optimization. This is an iterative way to optimize for multiple frequencies. The first step is to optimize over all frequencies in single. The second step is to choose the best single frequency and combine this with all others in double. This is possible at the moment. The next step is to save the best combined E-field from double and continue to combine this with other frequencies. Basically the optimization code for this already exists, double just needs to be able to take an E-field and a frequency instead of two frequencies. The disadvantage of this method is that it is very time-consuming, and all possible frequency combinations are not tested since every iteration adds on a new setting to a "locked" E-field.

2. Develop the polynomial code. The idea behind the polynomial optimization is to compute a polynomial of all frequencies at the same time, and find the minimal solution without iterations. In this way, all combinations are tested and there is no need for big loops. The downside is that multiple frequencies entails a very large number of variables (2*number of antennas*number of frequencies), which is very time-consuming. It also increases the risk of ending up in local minimums. The code for this exists only in parts, for example the conversion from a goal function to a C-executable polynomial, but not in totality. The folders Yggdrasil, Cpoly and Extrapolate on the GIT project contains the polynomial code. The code is not commented and we don't fully understand it. This has resulted in us being uncertain about some parts in optimization code, which is connected to the Cpoly content. It would be good to spend time to

gain better insight into how Cpoly operates and to make sure that the rest of the optimization connects properly to this.

### 3.2 Other optimization methods

There are other options of Matlab minimization functions than particle swarm. Other functions may have less randomness and therefore give more comparable results. If particle swarm is to be used, it would probably be a good idea to investigate the dependence of iterations and number of particles further and create a look up table. It would then also be of interest to study the randomness of particle swarm. How much of a difference in results can one expect from two identical optimizations?

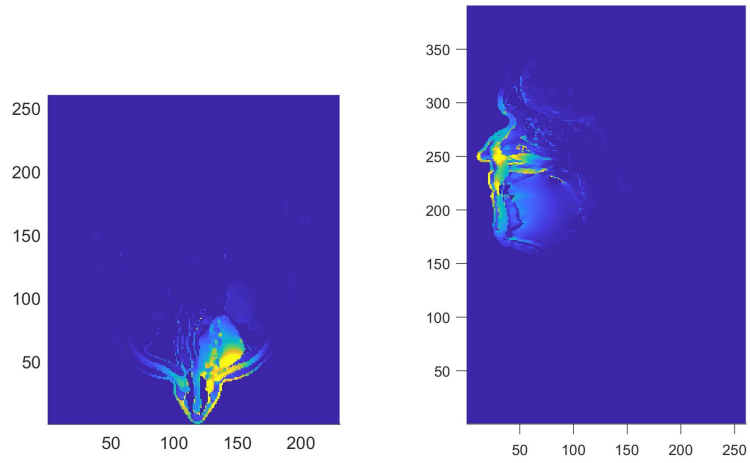
Other goal functions could also be tested. M2 is theoretically a very good goal function, but the extremely long time it takes to run with has to be shortened if M2 is going to be used. Overall, the time efficiency is a big problem and a lot of enhancement could probably be made here.

## A Obtained Results of Single Frequency Optimization

Results using the single frequency optimization are presented for the model NasalWerT. Results from tongue-salt are available in the document *Results Obtained During the Summer of 2017*.

Table 1: This table displays the values of HTQ, tumor coverage of 25, 50 and 75 %, the maximum % of all PLD that is in the tumor and the mean of normalized PLD in the model (the air outside the model is excluded). The values are calculated for an optimized PLD distribution for different frequencies (MHz) for the duke nasalWerT model. Particle swarm was set to 20 particles, 20 iterations and 10 stall iterations.

| **Frequency:** | **HTQ** | **TC25** | **TC50** | **TC75** | $\text{Max}_{\text{tumor}}\%$ | **Mean** |
|---|---|---|---|---|---|---|
| **400** | 0.9031 | 0.0053 | 0.0008 | 0.0001 | 1 | 0.0012 |
| **425** | 0.9101 | 0.0053 | 0.0007 | 0.0001 | 1 | 0.0012 |
| **450** | 0.9091 | 0.0060 | 0.0008 | 0.0001 | 1 | 0.0013 |
| **475** | 0.9208 | 0.0050 | 0.0007 | 0.0001 | 1 | 0.0012 |
| **500** | 0.9248 | 0.0050 | 0.0008 | 0.0001 | 1 | 0.0012 |
| **525** | 0.9424 | 0.0082 | 0.0013 | 0.0001 | 1 | 0.0015 |
| **550** | 0.9311 | 0.0058 | 0.0008 | 0.0001 | 1 | 0.0013 |
| **575** | 0.9322 | 0.0062 | 0.0008 | 0.0001 | 1 | 0.0013 |
| **600** | 0.9944 | 0.0067 | 0.0008 | 0.0001 | 0.9147 | 0.0013 |
| **625** | 0.9266 | 0.0077 | 0.0005 | 0 | 0.7354 | 0.0012 |
| **650** | 0.9291 | 0.0077 | 0.0005 | 0 | 0.7111 | 0.0012 |
| **675** | 0.9347 | 0.0076 | 0.0004 | 0 | 0.7077 | 0.0012 |
| **700** | 0.9467 | 0.0068 | 0.0003 | 0 | 0.6605 | 0.0012 |
| **725** | 0.9388 | 0.0076 | 0.0005 | 0 | 0.7085 | 0.0012 |
| **750** | 0.9505 | 0.0078 | 0.0007 | 0.0001 | 0.7997 | 0.0013 |
| **775** | 0.9466 | 0.0074 | 0.0004 | 0 | 0.6986 | 0.0012 |
| **800** | 0.9499 | 0.0078 | 0.0005 | 0 | 0.7407 | 0.0013 |
| **825** | 0.9460 | 0.0078 | 0.0005 | 0 | 0.7135 | 0.0013 |
| **850** | 0.9535 | 0.0067 | 0.0004 | 0 | 0.7385 | 0.0012 |
| **875** | 0.9456 | 0.0080 | 0.0005 | 0 | 0.7281 | 0.0013 |
| **900** | 0.9474 | 0.0084 | 0.0007 | 0.0001 | 0.7605 | 0.0013 |




(a) (b)

Figure 2: The images illustrate the PLD distribution for frequency 400 MHz, which gave the lowest HTQ value, 0.9031, for an optimization of nasalWerT using single frequency optimization. Subfigure 2a shows the distribution from above and 2b shows the PLD distribution from the side.

# B Obtained Results of Double Frequency Optimization

This section presents results from optimizations using two frequencies. First, the actual optimization with Double is shown in table 2 and then the results of a simpler version that combines already optimized E-fields are presented in table 3.

Table 2: This table displays the HTQ values for optimized PLD distributions using combined frequencies, with the Double Optimization.

| **[MHz]** | **400** | **425** | **450** | **475** | **500** |
|---|---|---|---|---|---|
| **400** | 0.8982 | 0.9100 | 0.9105 | 0.8996 | 0.9069 |
| **425** | 0.9048 | 0.9197 | 0.9087 | 0.9158 | 0.9109 |
| **450** | 0.9029 | 0.9134 | 0.9198 | 0.9170 | 0.9255 |
| **475** | 0.9045 | 0.9103 | 0.9169 | 0.9206 | 0.9189 |
| **500** | 0.9090 | 0.9101 | 0.9152 | 0.9179 | 0.9229 |

Table 3: This table displays the HTQ values for optimized PLD distributions when different frequencies have been combined but not optimized together, which can be used as a simpler and much quicker way to estimate the combination of multiple frequencies. The used optimization was Combine Single.

| [MHz] | **400** | **425** | **450** | **475** | **500** |
|---|---|---|---|---|---|
| **400** | | 0.9014 | 0.9014 | 0.9014 | 0.9014 |
| **425** | 0.9014 | | 0.9072 | 0.9084 | 0.9084 |
| **450** | 0.9014 | 0.9072 | | 0.9074 | 0.9074 |
| **475** | 0.9014 | 0.9084 | 0.9074 | | 0.9191 |
| **500** | 0.9014 | 0.9084 | 0.9074 | 0.9191 | |

The time shares that were used for both table 2 and table 3 are presented in tables 4 and 5.

Table 4: This table displays the time shares for optimized PLD distributions using combined frequencies, with the Double Optimization. The first time share in each cell shows the time for the frequency in the column and the second for the frequency in the row.

| [MHz] | **400** | **425** | **450** | **475** | **500** |
|---|---|---|---|---|---|
| **400** | 0.73 - 0.27 | 0 - 1 | 1 - 0 | 1 - 0 | 1 - 0 |
| **425** | 0.02 - 0.98 | 0 - 1 | 1 - 0 | 1 - 0 | 1 - 0 |
| **450** | 0 - 1 | 0.61 - 0.39 | 0.21 - 0.79 | 1 - 0 | 0.99 - 0.01 |
| **475** | 0 - 1 | 0 - 1 | 0.52 - 0.48 | 1 - 0 | 0.80 - 0.20 |
| **500** | 0 - 1 | 0 - 1 | 0 - 1 | 0 - 1 | 0.25 - 0.75 |

Table 5: This table displays the time shares for optimized PLD distributions when different frequencies have been combined but not optimized together, which can be used as a simpler and much quicker way to estimate the combination of multiple frequencies. The used optimization was Combine Single.

| [MHz] | **400** | **425** | **450** | **475** | **500** |
|---|---|---|---|---|---|
| **400** | | 1 - 0 | 1 - 0 | 1 - 0 | 1 - 0 |
| **425** | 0 - 1 | | 0.35 - 0.65 | 1 - 0 | 1 - 0 |
| **450** | 0 - 1 | 0.65 - 0.35 | | 1 - 0 | 1 - 0 |
| **475** | 0 - 1 | 0 - 1 | 0 - 1 | | 1 - 0 |
| **500** | 0 - 1 | 0 - 1 | 0 - 1 | 0 - 1 | |

# References

Chen, Stephen, James Montgomery, and Antonio Bolufé-Röhler (2015). *Measuring the curse of dimensionality and its effects on particle swarm optimization and differential evolution*. URL: https://link.springer.com/article/10.1007/s10489-014-0613-2.