Example: Photon Treatment Plan

Table of Contents

 1
Patient Data Import 1
Treatment Plan 3
Generate Beam Geometry STF 4
Dose Calculation 5
Inverse Optimization for IMRT 6
Plot the Resulting Dose Slice 13
Now let's create another treatment plan but this time use a coarser beam spacing. 14
Visual Comparison of results 23
Obtain dose statistics 26

In this example we will show (i) how to load patient data into matRad (ii) how to setup a photon dose calculation and (iii) how to inversely optimize beamlet intensities (iv) how to visually and quantitatively evaluate the result

Patient Data Import

Let's begin with a clear Matlab environment. Then, import the TG119 phantom into your workspace. The phantom is comprised of a 'ct' and 'cst' structure defining the CT images and the structure set. Make sure the matRad root directory with all its subdirectories is added to the Matlab search path.

```
clc,clear,close all;
load('TG119.mat');
```

The file TG119.mat contains two Matlab variables. Let's check what we have just imported. First, the 'ct' variable comprises the ct cube along

```
%with some meta information describing properties of the ct cube (cube
% dimensions, resolution, number of CT scenarios). Please note that
%multiple ct cubes (e.g. 4D CT) can be stored in the cell array
 ct.cube{}
display(ct);
```

```
ct =

  struct with fields:

           cube: {[167×167×129 double]}
     resolution: [1×1 struct]
        cubeDim: [167 167 129]
    numOfCtScen: 1
```

The 'cst' cell array defines volumes of interests along with information required for optimization. Each row belongs to one certain volume of interest (VOI), whereas each column defines different properties. Specifically, the second and third column show the name and the type of the structure. The type can be set to OAR, TARGET or IGNORED. The fourth column contains a linear index vector that lists all voxels belonging to a certain VOI.

```
display(cst);
```

```
cst =

  3×6 cell array

  Columns 1 through 5

    [0]    'BODY'           'OAR'       {1×1 cell}    [1×1 struct]
    [1]    'Core'           'OAR'       {1×1 cell}    [1×1 struct]
    [2]    'OuterTarget'    'TARGET'    {1×1 cell}    [1×1 struct]

  Column 6

    [1×1 struct]
    [1×1 struct]
    [1×1 struct]
```

The fifth column represents meta parameters for optimization. The overlap priority is used to resolve ambiguities of overlapping structures (voxels belonging to multiple structures will only be assigned to the VOI(s) with the highest overlap priority, i.e.. the lowest value). The parameters alphaX and betaX correspond to the tissue's photon-radiosensitivity parameter of the linear quadratic model. These parameter are required for biological treatment planning using a variable RBE. Let's output the meta optimization parameter of the target, which is stored in the thrid row:

```
ixTarget = 3;
display(cst{ixTarget,5});
```

```
    TissueClass: 1
         alphaX: 0.1000
          betaX: 0.0500
       Priority: 1
        Visible: 1
```

The sixth column contains optimization information such as objectives and constraints which are required to calculate the objective function value. Please note, that multiple objectives/constraints can be defined for individual structures. Here, we have defined a squared deviation objective making it 'expensive/costly' for the optimizer to over- and underdose the target structure (both are equally important).

```
display(cst{ixTarget,6});
```

```
          type: 'square deviation'
       penalty: 1000
          dose: 50
           EUD: NaN
        volume: NaN
    robustness: 'none'
```

Treatment Plan

The next step is to define your treatment plan labeled as 'pln'. This matlab structure requires input from the treatment planner and defines the most important cornerstones of your treatment plan.

First of all, we need to define what kind of radiation modality we would like to use. Possible values are photons, protons or carbon. In this case we want to use photons. Then, we need to define a treatment machine to correctly load the corresponding base data. matRad includes base data for generic photon linear accelerator called 'Generic'. By this means matRad will look for 'photons_Generic.mat' in our root directory and will use the data provided in there for dose calculation

```
pln.radiationMode = 'photons';
pln.machine       = 'Generic';
```

Define the flavor of optimization along with the quantity that should be used for optimization. Possible values are (none: physical optimization; const_RBExD: constant RBE of 1.1; LEMIV_effect: effect-based optimization; LEMIV_RBExD: optimization of RBE-weighted dose. As we are using photons, we simply set the parameter to 'none' thereby indicating the physical dose should be optimized.

```
pln.bioOptimization = 'none';
```

Now we have to set some beam parameters. We can define multiple beam angles for the treatment and pass these to the plan as a vector. matRad will then interpret the vector as multiple beams. In this case, we define linear spaced beams from 0 degree to 359 degree in 40 degree steps. This results in 9 beams. All corresponding couch angles are set to 0 at this point. Moreover, we set the bixelWidth to 5, which results in a beamlet size of 5 x 5 mm in the isocenter plane. The number of fractions is set to 30. Internally, matRad considers the fraction dose for optimization, however, objetives and constraints are defined for the entire treatment.

```
pln.gantryAngles    = [0:40:359];
pln.couchAngles     = zeros(1,numel(pln.gantryAngles));
pln.bixelWidth      = 5;
pln.numOfFractions  = 30;
```

Obtain the number of beams and voxels from the existing variables and calculate the iso-center which is per default the center of gravity of all target voxels.

```
pln.numOfBeams      = numel(pln.gantryAngles);
pln.numOfVoxels     = prod(ct.cubeDim);
pln.voxelDimensions = ct.cubeDim;
pln.isoCenter       = ones(pln.numOfBeams,1) *
 matRad_getIsoCenter(cst,ct,0);
```

Enable sequencing and disable direct aperture optimization (DAO) for now. A DAO optimization is shown in a seperate example.

```
pln.runSequencing = 1;
pln.runDAO        = 0;
```

and et voila our treatment plan structure is ready. Lets have a look:

```
display(pln);
```

```
pln =

  struct with fields:

      radiationMode: 'photons'
            machine: 'Generic'
    bioOptimization: 'none'
       gantryAngles: [0 40 80 120 160 200 240 280 320]
        couchAngles: [0 0 0 0 0 0 0 0 0]
         bixelWidth: 5
     numOfFractions: 30
         numOfBeams: 9
        numOfVoxels: 3597681
    voxelDimensions: [167 167 129]
          isoCenter: [9×3 double]
      runSequencing: 1
             runDAO: 0
```

Generate Beam Geometry STF

The steering file struct comprises the complete beam geometry along with ray position, pencil beam positions and energies, source to axis distance (SAD) etc.

```
stf = matRad_generateStf(ct,cst,pln);
```

```
matRad: Generating stf struct... Progress: 100.00 %
```

Let's display the beam geometry information of the 6th beam

```
display(stf(6));
```

```
          gantryAngle: 200
           couchAngle: 0
           bixelWidth: 5
        radiationMode: 'photons'
                  SAD: 1000
            isoCenter: [250.7385 236.3393 162.6736]
            numOfRays: 305
                  ray: [1×305 struct]
       sourcePoint_bev: [0 -1000 0]
          sourcePoint: [-342.0201 939.6926 0]
    numOfBixelsPerRay: [1×305 double]
     totalNumOfBixels: 305
```

Dose Calculation

Let's generate dosimetric information by pre-computing dose influence matrices for unit beamlet intensities. Having dose influences available allows subsequent inverse optimization.

```
dij       = matRad_calcPhotonDose(ct,stf,pln,cst);
```

```
matRad: Photon dose calculation...
Beam 1 of 9:
matRad: calculate radiological depth cube...done
                   SSD = 939mm
matRad: Uniform primary photon fluence -> pre-compute kernel
 convolution for SSD = 939 mm ...
Progress: 100.00 %
Beam 2 of 9:
matRad: calculate radiological depth cube...done
                   SSD = 921mm
matRad: Uniform primary photon fluence -> pre-compute kernel
 convolution for SSD = 921 mm ...
Progress: 100.00 %
Beam 3 of 9:
matRad: calculate radiological depth cube...done
                   SSD = 848mm
matRad: Uniform primary photon fluence -> pre-compute kernel
 convolution for SSD = 848 mm ...
Progress: 100.00 %
Beam 4 of 9:
matRad: calculate radiological depth cube...done
                   SSD = 827mm
matRad: Uniform primary photon fluence -> pre-compute kernel
 convolution for SSD = 827 mm ...
Progress: 100.00 %
Beam 5 of 9:
matRad: calculate radiological depth cube...done
                   SSD = 902mm
matRad: Uniform primary photon fluence -> pre-compute kernel
 convolution for SSD = 902 mm ...
Progress: 100.00 %
Beam 6 of 9:
matRad: calculate radiological depth cube...done
                   SSD = 905mm
matRad: Uniform primary photon fluence -> pre-compute kernel
 convolution for SSD = 905 mm ...
Progress: 100.00 %
Beam 7 of 9:
matRad: calculate radiological depth cube...done
                   SSD = 827mm
matRad: Uniform primary photon fluence -> pre-compute kernel
 convolution for SSD = 827 mm ...
Progress: 100.00 %
Beam 8 of 9:
matRad: calculate radiological depth cube...done
                   SSD = 847mm
```

```
matRad: Uniform primary photon fluence -> pre-compute kernel
 convolution for SSD = 847 mm ...
Progress: 100.00 %
Beam 9 of 9:
matRad: calculate radiological depth cube...done
                   SSD = 921mm
matRad: Uniform primary photon fluence -> pre-compute kernel
 convolution for SSD = 921 mm ...
Progress: 100.00 %
```

Inverse Optimization for IMRT

The goal of the fluence optimization is to find a set of beamlet/pencil beam weights which yield the best possible dose distribution according to the clinical objectives and constraints underlying the radiation treatment. Once the optimization has finished, trigger once the GUI to visualize the optimized dose cubes.

```
resultGUI = matRad_fluenceOptimization(dij,cst,pln);
matRadGUI;
```

```

******************************************************************************
This program contains Ipopt, a library for large-scale nonlinear
 optimization.
 Ipopt is released as open source code under the Eclipse Public
 License (EPL).
         For more information visit http://projects.coin-or.org/Ipopt
******************************************************************************

This is Ipopt version 3.11.8, running with linear solver ma57.

Number of nonzeros in equality constraint Jacobian...:        0
Number of nonzeros in inequality constraint Jacobian.:        0
Number of nonzeros in Lagrangian Hessian.............:        0

Total number of variables............................:     2471
                     variables with only lower bounds:     2471
                variables with lower and upper bounds:        0
                     variables with only upper bounds:        0
Total number of equality constraints.................:        0
Total number of inequality constraints...............:        0
        inequality constraints with only lower bounds:        0
   inequality constraints with lower and upper bounds:        0
        inequality constraints with only upper bounds:        0

iter    objective    inf_pr   inf_du lg(mu)  ||d||  lg(rg) alpha_du
 alpha_pr  ls
   0 1.8045461e+002 0.00e+000 5.59e+000   0.0 0.00e+000    -  0.00e
+000 0.00e+000   0
   1 7.7598883e+001 0.00e+000 4.95e+000  -0.3 9.78e-001    -
 9.83e-001 2.60e-001f  1
   2 7.7326246e+001 0.00e+000 4.18e+000  -6.6 1.46e-001    -  1.00e
+000 1.00e+000f  1
   3 1.2870774e+001 0.00e+000 1.33e+000  -1.6 7.39e-002    -
 9.98e-001 1.00e+000f  1
```

```
   4 9.8604640e+000 0.00e+000 6.57e-001  -2.4 1.71e-002    -
 9.97e-001 1.00e+000f  1
   5 4.1020373e+000 0.00e+000 4.63e-001  -2.8 8.09e-002    -  1.00e
+000 1.00e+000f  1
   6 3.0684212e+000 0.00e+000 5.98e-001  -3.6 5.52e-002    -  1.00e
+000 1.00e+000f  1
   7 2.1642016e+000 0.00e+000 1.99e-001  -4.4 1.40e-002    -  1.00e
+000 1.00e+000f  1
   8 2.0384449e+000 0.00e+000 1.59e-001  -5.7 1.64e-002    -  1.00e
+000 3.98e-001f  1
   9 1.9318526e+000 0.00e+000 1.45e-001  -7.0 2.77e-002    -  1.00e
+000 2.46e-001f  1
iter    objective    inf_pr   inf_du lg(mu)  ||d||  lg(rg) alpha_du
 alpha_pr  ls
  10 1.8196079e+000 0.00e+000 1.14e-001  -7.9 3.86e-002    -  1.00e
+000 2.05e-001f  1
  11 1.6743000e+000 0.00e+000 1.15e-001  -3.8 4.68e-002    -  1.00e
+000 2.32e-001f  1
  12 1.4560687e+000 0.00e+000 1.39e+000  -3.1 1.93e-001    -
 8.92e-001 6.37e-001f  1
  13 1.5434436e+000 0.00e+000 8.98e-001  -2.5 5.82e-002    -
 9.39e-001 1.00e+000f  1
  14 1.3971002e+000 0.00e+000 1.61e-001  -2.6 5.37e-002    -
 9.78e-001 1.00e+000f  1
  15 1.2458002e+000 0.00e+000 1.43e-001  -3.6 3.53e-002    -  1.00e
+000 1.00e+000f  1
  16 1.1074278e+000 0.00e+000 1.56e-001  -4.8 3.53e-002    -  1.00e
+000 1.00e+000f  1
  17 1.0452536e+000 0.00e+000 7.28e-002  -5.7 2.32e-002    -  1.00e
+000 6.08e-001f  1
  18 1.0181998e+000 0.00e+000 5.47e-002 -11.0 1.97e-002    -
 6.66e-001 5.35e-001f  1
  19 9.9372978e-001 0.00e+000 5.76e-002  -3.8 2.29e-002    -
 5.67e-001 1.00e+000f  1
iter    objective    inf_pr   inf_du lg(mu)  ||d||  lg(rg) alpha_du
 alpha_pr  ls
  20 9.7579607e-001 0.00e+000 6.36e-002  -4.7 2.01e-002    -
 8.23e-001 1.00e+000f  1
  21 9.6852344e-001 0.00e+000 5.52e-002  -5.5 1.42e-002    -  1.00e
+000 3.14e-001f  1
  22 9.4314677e-001 0.00e+000 5.46e-002  -6.6 2.90e-002    -  1.00e
+000 5.93e-001f  1
  23 9.3625841e-001 0.00e+000 5.86e-002  -6.6 2.78e-002    -
 9.59e-001 1.70e-001f  1
  24 9.2561662e-001 0.00e+000 7.24e-002  -7.1 4.13e-002    -  1.00e
+000 1.71e-001f  1
  25 9.1453676e-001 0.00e+000 1.41e-001  -4.7 4.51e-002    -
 9.78e-001 1.59e-001f  1
  26 9.0065020e-001 0.00e+000 1.00e-001  -5.0 3.83e-002    -
 9.17e-001 2.46e-001f  1
  27 8.9586836e-001 0.00e+000 4.84e-002 -11.0 4.69e-002    -
 6.34e-001 7.88e-002f  1
  28 8.8409851e-001 0.00e+000 2.87e-001  -3.7 2.71e-002    -
 9.75e-001 1.00e+000f  1
```

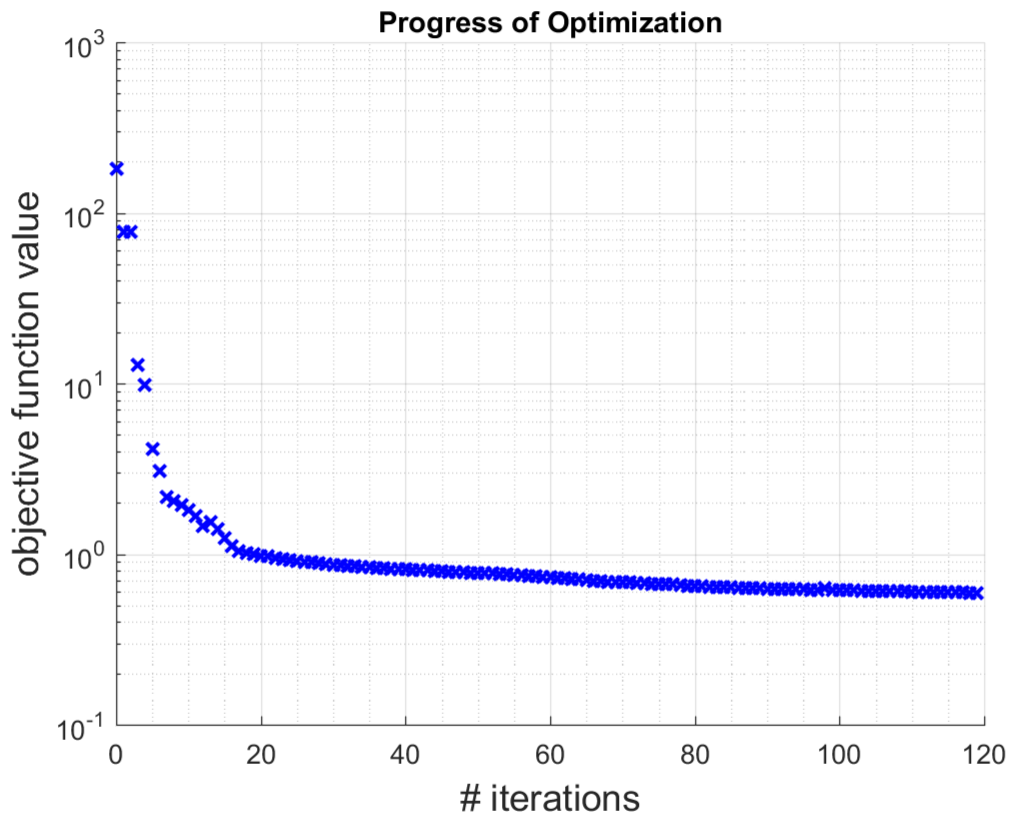
```
  29 8.7439333e-001 0.00e+000 3.49e-002  -4.3 2.69e-002    -  1.00e
+000 5.00e-001f  2
iter    objective    inf_pr   inf_du lg(mu)  ||d||  lg(rg) alpha_du
 alpha_pr  ls
  30 8.6577408e-001 0.00e+000 2.63e-002  -6.0 7.90e-003    -  1.00e
+000 1.00e+000f  1
  31 8.5754951e-001 0.00e+000 2.64e-002  -7.2 8.45e-003    -  1.00e
+000 9.32e-001f  1
  32 8.5334149e-001 0.00e+000 3.49e-002  -8.1 1.59e-002    -  1.00e
+000 2.36e-001f  1
  33 8.4695980e-001 0.00e+000 4.09e-002  -4.7 1.96e-002    -
 6.66e-001 2.33e-001f  1
  34 8.4238979e-001 0.00e+000 9.69e-002  -4.6 1.41e-002    -
 8.17e-001 1.74e-001f  1
  35 8.3664693e-001 0.00e+000 1.17e-001  -4.0 3.21e-002    -
 5.17e-001 7.81e-001f  1
  36 8.3063558e-001 0.00e+000 6.95e-002  -4.8 9.04e-003    -
 9.57e-001 3.92e-001f  1
  37 8.2738162e-001 0.00e+000 8.14e-002 -10.7 9.19e-003    -
 8.17e-001 2.28e-001f  1
  38 8.2098753e-001 0.00e+000 1.15e-001  -6.3 1.40e-002    -
 9.77e-001 3.16e-001f  1
  39 8.1396938e-001 0.00e+000 1.14e-001  -4.9 1.77e-002    -
 9.48e-001 3.15e-001f  1
iter    objective    inf_pr   inf_du lg(mu)  ||d||  lg(rg) alpha_du
 alpha_pr  ls
  40 8.1587066e-001 0.00e+000 1.74e-001  -4.0 2.61e-002    -
 6.13e-001 1.00e+000f  1
  41 8.0840509e-001 0.00e+000 5.49e-002  -4.2 7.99e-003    -
 9.41e-001 8.85e-001f  1
  42 8.0357978e-001 0.00e+000 4.26e-002  -4.7 8.11e-003    -  1.00e
+000 8.43e-001f  1
  43 8.0206491e-001 0.00e+000 1.06e-001  -6.7 1.08e-002    -
 8.41e-001 2.02e-001f  1
  44 7.9863123e-001 0.00e+000 7.94e-002  -6.7 2.15e-002    -
 9.22e-001 2.17e-001f  1
  45 7.9269927e-001 0.00e+000 1.09e-001  -6.9 2.41e-002    -  1.00e
+000 3.19e-001f  1
  46 7.8923575e-001 0.00e+000 5.08e-002  -6.8 2.95e-002    -  1.00e
+000 1.71e-001f  1
  47 7.8651051e-001 0.00e+000 6.88e-002  -7.5 3.51e-002    -
 7.59e-001 1.15e-001f  1
  48 7.8452685e-001 0.00e+000 8.39e-002  -6.6 2.11e-002    -
 9.60e-001 9.15e-002f  1
  49 7.7839858e-001 0.00e+000 1.64e-001  -5.7 1.98e-002    -  1.00e
+000 2.81e-001f  1
iter    objective    inf_pr   inf_du lg(mu)  ||d||  lg(rg) alpha_du
 alpha_pr  ls
  50 7.7575231e-001 0.00e+000 6.25e-002  -6.1 2.29e-002    -
 6.79e-001 1.10e-001f  1
  51 7.7383229e-001 0.00e+000 1.18e-001  -6.0 1.73e-002    -  1.00e
+000 1.16e-001f  1
  52 7.6969351e-001 0.00e+000 3.90e-002  -6.0 3.14e-002    -
 9.87e-001 1.61e-001f  1
```

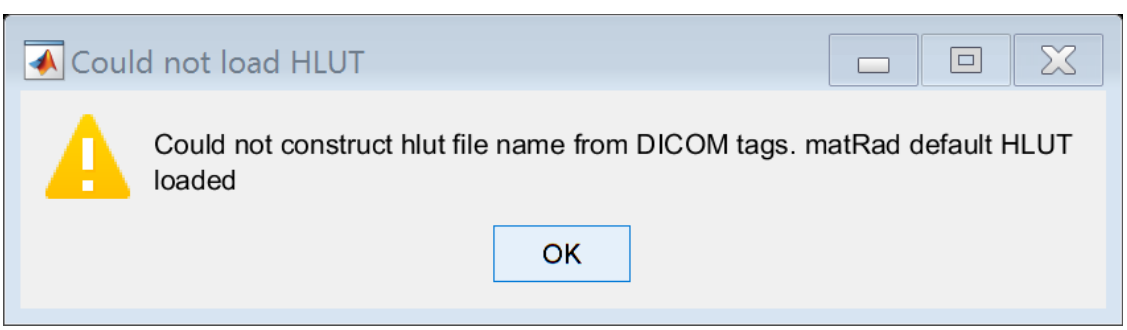
```
  53 7.6448967e-001 0.00e+000 1.65e-001  -4.6 1.85e-002    -
 7.36e-001 3.98e-001f  1
  54 7.5934547e-001 0.00e+000 3.90e-002  -4.7 1.90e-002    -
 6.84e-001 4.95e-001f  1
  55 7.5550676e-001 0.00e+000 5.39e-002  -4.5 1.55e-002    -
 5.91e-001 5.43e-001f  1
  56 7.5194768e-001 0.00e+000 4.36e-002  -5.0 1.06e-002    -
 7.98e-001 6.92e-001f  1
  57 7.4773768e-001 0.00e+000 4.39e-002  -4.9 3.02e-002    -
 6.49e-001 2.81e-001f  1
  58 7.4408967e-001 0.00e+000 2.87e-002  -4.6 1.74e-002    -
 6.94e-001 3.33e-001f  1
  59 7.3362734e-001 0.00e+000 3.68e-002  -4.5 2.43e-002    -
 5.23e-001 1.00e+000f  1
iter    objective    inf_pr   inf_du lg(mu)  ||d||  lg(rg) alpha_du
 alpha_pr  ls
  60 7.3170654e-001 0.00e+000 4.89e-002  -4.0 1.60e-002    -
 5.64e-001 1.00e+000f  1
  61 7.2590677e-001 0.00e+000 2.02e-002  -4.7 3.29e-003    -  1.00e
+000 1.00e+000f  1
  62 7.2089200e-001 0.00e+000 3.01e-002  -5.1 6.01e-003    -  1.00e
+000 1.00e+000f  1
  63 7.1533768e-001 0.00e+000 3.26e-002  -5.7 9.98e-003    -  1.00e
+000 7.03e-001f  1
  64 7.1126148e-001 0.00e+000 3.27e-002  -6.3 2.01e-002    -  1.00e
+000 2.27e-001f  1
  65 7.0628573e-001 0.00e+000 4.28e-002  -4.4 2.38e-002    -  1.00e
+000 3.60e-001f  1
  66 6.9743317e-001 0.00e+000 7.33e-002  -4.4 2.57e-002    -
 9.12e-001 5.58e-001f  1
  67 6.9456481e-001 0.00e+000 8.32e-002  -4.4 1.33e-002    -
 7.35e-001 5.97e-001f  1
  68 6.8695183e-001 0.00e+000 4.49e-002  -5.2 2.40e-002    -
 9.03e-001 7.47e-001f  1
  69 6.8768449e-001 0.00e+000 4.97e-002  -4.3 1.75e-002    -  1.00e
+000 1.00e+000f  1
iter    objective    inf_pr   inf_du lg(mu)  ||d||  lg(rg) alpha_du
 alpha_pr  ls
  70 6.8272321e-001 0.00e+000 4.19e-002  -4.3 5.03e-003    -
 9.24e-001 1.00e+000f  1
  71 6.8217298e-001 0.00e+000 3.56e-002  -4.1 1.17e-002    -
 6.25e-001 1.00e+000f  1
  72 6.7581007e-001 0.00e+000 2.35e-002  -4.9 1.02e-002    -  1.00e
+000 1.00e+000f  1
  73 6.7149815e-001 0.00e+000 2.13e-002  -5.5 1.40e-002    -  1.00e
+000 7.19e-001f  1
  74 6.6813549e-001 0.00e+000 3.06e-002  -5.2 1.49e-002    -  1.00e
+000 6.03e-001f  1
  75 6.6392516e-001 0.00e+000 2.85e-002  -4.9 1.31e-002    -  1.00e
+000 8.36e-001f  1
  76 6.6797723e-001 0.00e+000 5.53e-002  -4.3 1.42e-002    -
 7.88e-001 1.00e+000f  1
  77 6.6298270e-001 0.00e+000 3.47e-002  -4.4 2.86e-002    -
 9.19e-001 5.00e-001f  2
```

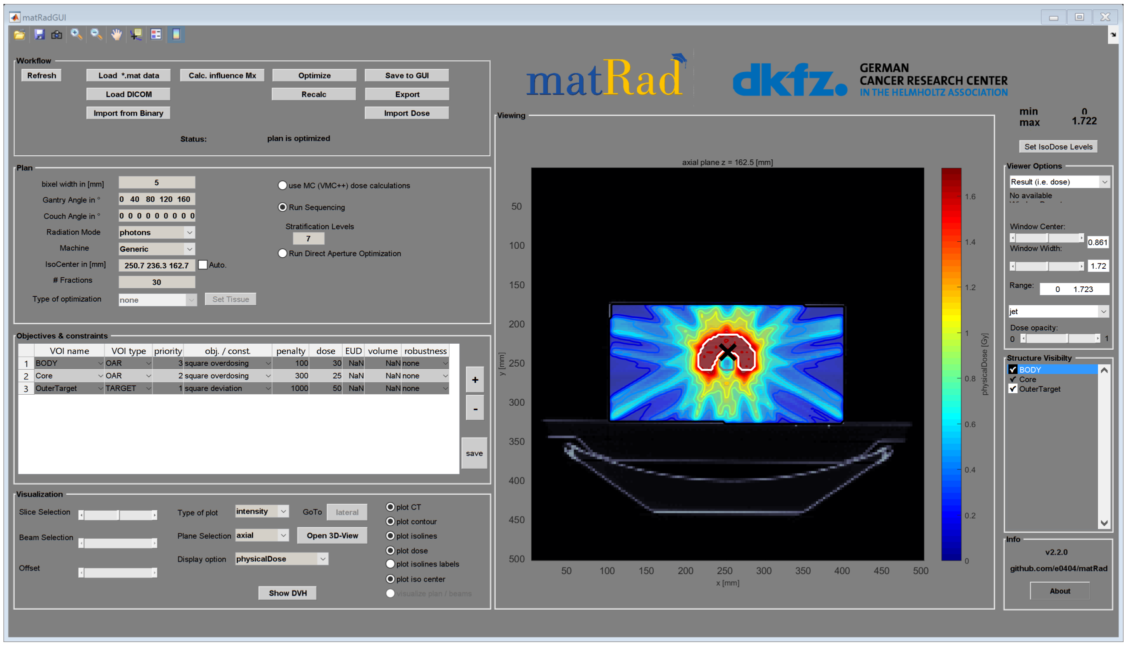
```
  78 6.5563258e-001 0.00e+000 1.95e-002  -4.6 1.32e-002    -  1.00e
+000 1.00e+000f  1
  79 6.5270764e-001 0.00e+000 2.42e-002  -5.0 1.51e-002    -  1.00e
+000 4.63e-001f  1
iter    objective    inf_pr   inf_du lg(mu)  ||d||  lg(rg) alpha_du
 alpha_pr  ls
  80 6.4760753e-001 0.00e+000 6.41e-002  -5.6 2.19e-002    -  1.00e
+000 7.10e-001f  1
  81 6.4588426e-001 0.00e+000 2.90e-002  -6.7 2.43e-002    -
 8.98e-001 2.29e-001f  1
  82 6.4242410e-001 0.00e+000 2.40e-002  -5.4 3.64e-002    -
 9.30e-001 3.36e-001f  1
  83 6.4182908e-001 0.00e+000 7.18e-002  -4.5 4.26e-002    -
 5.40e-001 1.00e+000f  1
  84 6.3752985e-001 0.00e+000 3.47e-002  -5.3 2.30e-002    -  1.00e
+000 5.59e-001f  1
  85 6.3728393e-001 0.00e+000 6.00e-002  -5.5 9.28e-003    -  1.00e
+000 4.95e-002f  1
  86 6.3532223e-001 0.00e+000 4.24e-002  -5.6 1.26e-002    -  1.00e
+000 2.12e-001f  1
  87 6.3274838e-001 0.00e+000 4.77e-002 -11.0 1.13e-002    -
 3.05e-001 2.81e-001f  1
  88 6.2975660e-001 0.00e+000 5.75e-002  -6.1 8.55e-003    -  1.00e
+000 3.36e-001f  1
  89 6.2893229e-001 0.00e+000 5.32e-002  -6.4 1.03e-002    -  1.00e
+000 9.63e-002f  1
iter    objective    inf_pr   inf_du lg(mu)  ||d||  lg(rg) alpha_du
 alpha_pr  ls
  90 6.2702846e-001 0.00e+000 9.56e-002  -6.5 1.40e-002    -
 5.92e-001 1.78e-001f  1
  91 6.2554939e-001 0.00e+000 7.98e-002  -7.0 1.39e-002    -
 8.85e-001 1.37e-001f  1
  92 6.2439157e-001 0.00e+000 1.01e-001  -7.1 1.44e-002    -
 9.21e-001 1.10e-001f  1
  93 6.2143361e-001 0.00e+000 3.46e-002  -7.4 2.21e-002    -  1.00e
+000 1.94e-001f  1
  94 6.2072986e-001 0.00e+000 1.41e-001  -7.9 1.65e-002    -
 6.44e-001 5.87e-002f  1
  95 6.1994622e-001 0.00e+000 2.93e-002  -8.4 2.81e-002    -
 8.36e-001 4.04e-002f  1
  96 6.1707890e-001 0.00e+000 1.19e-001  -5.5 3.05e-002    -
 8.85e-001 1.40e-001f  1
  97 6.1611984e-001 0.00e+000 8.47e-002  -5.9 2.04e-002    -
 3.38e-001 7.27e-002f  1
  98 6.3130484e-001 0.00e+000 6.88e-002  -4.3 1.94e-002    -
 6.48e-001 1.00e+000f  1
  99 6.1698414e-001 0.00e+000 2.93e-002  -4.5 6.63e-003    -  1.00e
+000 9.00e-001f  1
iter    objective    inf_pr   inf_du lg(mu)  ||d||  lg(rg) alpha_du
 alpha_pr  ls
 100 6.1540333e-001 0.00e+000 4.40e-002  -4.5 1.73e-003    -
 8.47e-001 1.00e+000f  1
 101 6.1342299e-001 0.00e+000 5.61e-002  -5.3 5.12e-003    -  1.00e
+000 5.33e-001f  1
```

```
 102 6.1105066e-001 0.00e+000 6.35e-002  -6.3 7.21e-003    -  1.00e
+000 3.72e-001f  1
 103 6.0861177e-001 0.00e+000 8.11e-002  -6.9 1.05e-002    -  1.00e
+000 3.15e-001f  1
 104 6.0727831e-001 0.00e+000 5.46e-002  -7.7 1.35e-002    -  1.00e
+000 1.68e-001f  1
 105 6.0596102e-001 0.00e+000 4.78e-002  -5.4 8.09e-003    -
 5.82e-001 2.84e-001f  1
 106 6.1058986e-001 0.00e+000 1.11e-001  -4.4 4.56e-002    -
 3.37e-001 1.00e+000f  1
 107 6.0806816e-001 0.00e+000 1.87e-002  -4.8 1.25e-002    -
 9.59e-001 3.04e-001f  1
 108 6.0436148e-001 0.00e+000 4.41e-002  -4.8 7.04e-003    -
 5.96e-001 7.41e-001f  1
 109 6.0342860e-001 0.00e+000 4.88e-002  -5.7 6.77e-003    -
 7.39e-001 1.91e-001f  1
iter    objective    inf_pr   inf_du lg(mu)  ||d||  lg(rg) alpha_du
 alpha_pr  ls
 110 6.0186463e-001 0.00e+000 3.39e-002  -6.7 8.13e-003    -
 6.93e-001 2.73e-001f  1
 111 6.0045597e-001 0.00e+000 4.32e-002  -6.5 7.07e-003    -
 6.99e-001 2.92e-001f  1
 112 6.0026600e-001 0.00e+000 4.58e-002  -7.2 1.07e-002    -
 7.78e-001 3.07e-002f  1
 113 5.9818168e-001 0.00e+000 3.93e-002  -6.8 1.83e-002    -
 8.71e-001 2.27e-001f  1
 114 5.9706058e-001 0.00e+000 4.91e-002  -5.9 1.97e-002    -
 3.83e-001 1.23e-001f  1
 115 5.9661329e-001 0.00e+000 6.72e-002 -11.0 1.08e-002    -
 4.48e-001 9.16e-002f  1
 116 5.9513604e-001 0.00e+000 3.41e-002  -6.2 2.54e-002    -
 5.27e-001 1.42e-001f  1
 117 5.9463027e-001 0.00e+000 4.66e-002  -6.3 9.45e-003    -
 5.55e-001 1.14e-001f  1
 118 5.9373395e-001 0.00e+000 3.52e-002  -6.2 1.57e-002    -
 3.45e-001 1.24e-001f  1
 119 5.9312237e-001 0.00e+000 3.78e-002  -5.9 1.60e-002    -
 4.88e-001 8.34e-002f  1

Number of Iterations....: 119

                                   (scaled)                 (unscaled)
Objective...............:  5.9312236540050356e-001
 5.9312236540050356e-001
Dual infeasibility......:  3.7841495524588778e-002
 3.7841495524588778e-002
Constraint violation....:  0.0000000000000000e+000
 0.0000000000000000e+000
Complementarity.........:  6.6217345445157698e-006
 6.6217345445157698e-006
Overall NLP error.......:  3.7841495524588778e-002
 3.7841495524588778e-002
```

```
Number of objective function evaluations             = 130
Number of objective gradient evaluations             = 120
Number of equality constraint evaluations            = 0
Number of inequality constraint evaluations          = 0
Number of equality constraint Jacobian evaluations   = 0
Number of inequality constraint Jacobian evaluations = 0
Number of Lagrangian Hessian evaluations             = 0
Total CPU secs in IPOPT (w/o function evaluations)   =      9.703
Total CPU secs in NLP function evaluations           =    111.691

EXIT: Solved To Acceptable Level.
Calculating final cubes...
Warning: matRad default HLUT loaded
Reconversion of HU values could not be done because HLUT is not
 bijective.
```



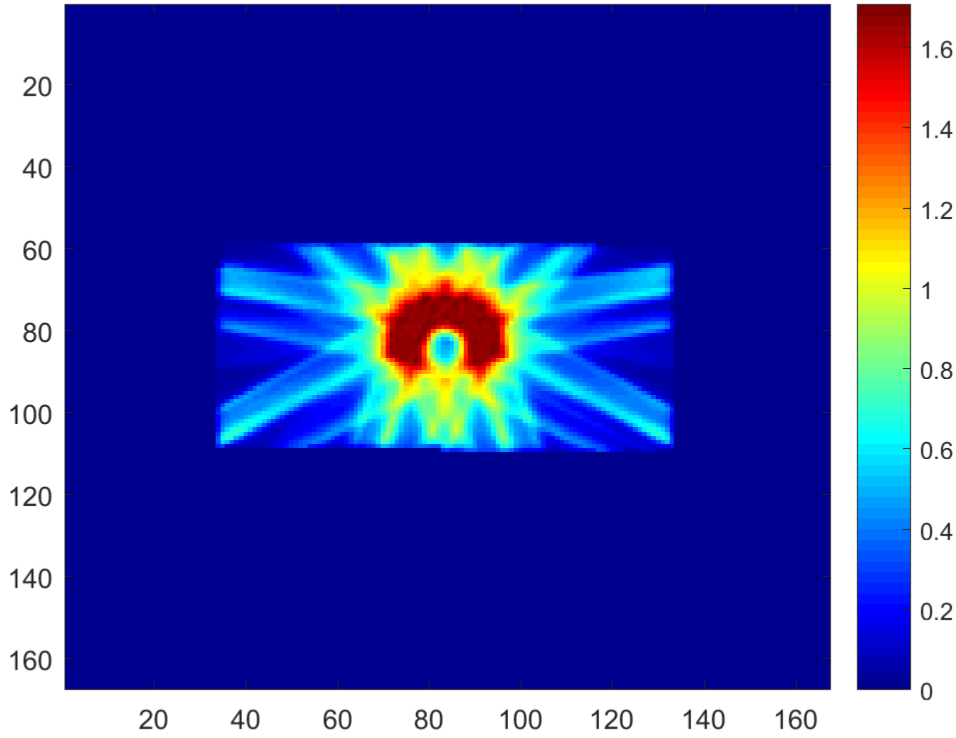




Plot the Resulting Dose Slice

Let's plot the transversal iso-center dose slice

```
slice = round(pln.isoCenter(1,3)./ct.resolution.z);
figure
imagesc(resultGUI.physicalDose(:,:,slice)),colorbar, colormap(jet);
```



Now let's create another treatment plan but this time use a coarser beam spacing.

Instead of 40 degree spacing use a 50 degree geantry beam spacing

```
pln.gantryAngles = [0:50:359];
pln.couchAngles  = zeros(1,numel(pln.gantryAngles));
pln.numOfBeams   = numel(pln.gantryAngles);
stf              = matRad_generateStf(ct,cst,pln);
pln.isoCenter    = stf.isoCenter;
dij              = matRad_calcPhotonDose(ct,stf,pln,cst);
resultGUI_coarse = matRad_fluenceOptimization(dij,cst,pln);
```

```
matRad: Generating stf struct... Progress: 100.00 %
matRad: Photon dose calculation...
Beam 1 of 8:
matRad: calculate radiological depth cube...done
                   SSD = 939mm
matRad: Uniform primary photon fluence -> pre-compute kernel
 convolution for SSD = 939 mm ...
Progress: 100.00 %
Beam 2 of 8:
matRad: calculate radiological depth cube...done
                   SSD = 905mm
```

```
matRad: Uniform primary photon fluence -> pre-compute kernel
 convolution for SSD = 905 mm ...
Progress: 100.00 %
Beam 3 of 8:
matRad: calculate radiological depth cube...done
                   SSD = 848mm
matRad: Uniform primary photon fluence -> pre-compute kernel
 convolution for SSD = 848 mm ...
Progress: 100.00 %
Beam 4 of 8:
matRad: calculate radiological depth cube...done
                   SSD = 894mm
matRad: Uniform primary photon fluence -> pre-compute kernel
 convolution for SSD = 894 mm ...
Progress: 100.00 %
Beam 5 of 8:
matRad: calculate radiological depth cube...done
                   SSD = 905mm
matRad: Uniform primary photon fluence -> pre-compute kernel
 convolution for SSD = 905 mm ...
Progress: 100.00 %
Beam 6 of 8:
matRad: calculate radiological depth cube...done
                   SSD = 840mm
matRad: Uniform primary photon fluence -> pre-compute kernel
 convolution for SSD = 840 mm ...
Progress: 100.00 %
Beam 7 of 8:
matRad: calculate radiological depth cube...done
                   SSD = 878mm
matRad: Uniform primary photon fluence -> pre-compute kernel
 convolution for SSD = 878 mm ...
Progress: 100.00 %
Beam 8 of 8:
matRad: calculate radiological depth cube...done
                   SSD = 938mm
matRad: Uniform primary photon fluence -> pre-compute kernel
 convolution for SSD = 938 mm ...
Progress: 100.00 %

******************************************************************************
This program contains Ipopt, a library for large-scale nonlinear
 optimization.
 Ipopt is released as open source code under the Eclipse Public
 License (EPL).
         For more information visit http://projects.coin-or.org/Ipopt
******************************************************************************

This is Ipopt version 3.11.8, running with linear solver ma57.

Number of nonzeros in equality constraint Jacobian...:        0
Number of nonzeros in inequality constraint Jacobian.:        0
Number of nonzeros in Lagrangian Hessian.............:        0
```

```
Total number of variables............................:     2207
                     variables with only lower bounds:     2207
                variables with lower and upper bounds:        0
                     variables with only upper bounds:        0
Total number of equality constraints.................:        0
Total number of inequality constraints...............:        0
        inequality constraints with only lower bounds:        0
   inequality constraints with lower and upper bounds:        0
        inequality constraints with only upper bounds:        0

iter    objective    inf_pr   inf_du lg(mu)  ||d||  lg(rg) alpha_du
 alpha_pr  ls
   0 1.6835152e+002 0.00e+000 5.54e+000   0.0 0.00e+000    -  0.00e
+000 0.00e+000   0
   1 9.0229894e+001 0.00e+000 5.64e+000  -0.3 1.03e+000    -
 9.77e-001 2.67e-001f  1
   2 7.2225245e+001 0.00e+000 4.15e+000  -6.6 1.45e-001    -  1.00e
+000 1.00e+000f  1
   3 1.5886846e+001 0.00e+000 2.49e+000  -1.6 6.87e-002    -
 9.99e-001 1.00e+000f  1
   4 1.2711254e+001 0.00e+000 6.60e-001  -1.7 2.21e-002    -
 9.28e-001 1.00e+000f  1
   5 7.5346485e+000 0.00e+000 4.25e-001  -2.0 5.31e-002    -
 9.98e-001 1.00e+000f  1
   6 4.3820387e+000 0.00e+000 3.54e-001  -2.8 6.10e-002    -  1.00e
+000 1.00e+000f  1
   7 3.2914364e+000 0.00e+000 5.76e-001  -3.4 8.20e-002    -  1.00e
+000 1.00e+000f  1
   8 2.3991188e+000 0.00e+000 1.84e-001  -3.8 2.75e-002    -  1.00e
+000 8.41e-001f  1
   9 2.2613266e+000 0.00e+000 1.46e-001  -4.8 9.15e-003    -  1.00e
+000 1.00e+000f  1
iter    objective    inf_pr   inf_du lg(mu)  ||d||  lg(rg) alpha_du
 alpha_pr  ls
  10 2.1628239e+000 0.00e+000 1.46e-001  -5.8 2.27e-002    -  1.00e
+000 3.33e-001f  1
  11 2.0423020e+000 0.00e+000 1.37e-001  -4.6 4.60e-002    -  1.00e
+000 2.25e-001f  1
  12 1.9278596e+000 0.00e+000 5.51e-001  -4.0 5.47e-002    -  1.00e
+000 1.98e-001f  1
  13 1.8476477e+000 0.00e+000 8.75e-001  -2.9 1.32e-001    -
 6.38e-001 1.00e+000f  1
  14 1.7538306e+000 0.00e+000 1.58e-001  -4.2 2.31e-002    -
 9.90e-001 1.00e+000f  1
  15 1.6968139e+000 0.00e+000 7.91e-002  -4.4 3.37e-002    -  1.00e
+000 1.00e+000f  1
  16 1.6593228e+000 0.00e+000 5.78e-002  -5.7 3.13e-002    -  1.00e
+000 6.94e-001f  1
  17 1.6351462e+000 0.00e+000 1.14e-001  -6.2 2.83e-002    -  1.00e
+000 2.83e-001f  1
  18 1.6021937e+000 0.00e+000 9.55e-002  -7.1 4.55e-002    -  1.00e
+000 2.10e-001f  1
  19 1.5710715e+000 0.00e+000 5.16e-002  -7.4 3.70e-002    -  1.00e
+000 1.75e-001f  1
```

```
iter    objective    inf_pr   inf_du lg(mu)  ||d||  lg(rg) alpha_du
 alpha_pr  ls
  20 1.5499955e+000 0.00e+000 2.04e-001  -4.2 5.32e-002    -
 9.31e-001 9.40e-002f  1
  21 1.4927035e+000 0.00e+000 4.58e-001  -3.7 9.24e-002    -
 8.51e-001 3.20e-001f  1
  22 1.5233347e+000 0.00e+000 4.25e-001  -3.3 7.05e-002    -
 8.15e-001 1.00e+000f  1
  23 1.4751530e+000 0.00e+000 1.02e-001  -3.3 1.23e-002    -  1.00e
+000 1.00e+000f  1
  24 1.4515605e+000 0.00e+000 9.01e-002  -4.0 1.94e-002    -
 9.99e-001 1.00e+000f  1
  25 1.4320297e+000 0.00e+000 1.03e-001  -5.3 2.17e-002    -  1.00e
+000 6.07e-001f  1
  26 1.4112400e+000 0.00e+000 6.73e-002  -5.8 2.85e-002    -  1.00e
+000 3.49e-001f  1
  27 1.4043540e+000 0.00e+000 1.38e-001  -5.2 1.59e-002    -
 7.77e-001 1.71e-001f  1
  28 1.3872250e+000 0.00e+000 5.48e-002  -5.6 3.60e-002    -  1.00e
+000 2.05e-001f  1
  29 1.3571939e+000 0.00e+000 9.36e-002  -5.2 5.12e-002    -  1.00e
+000 3.43e-001f  1
iter    objective    inf_pr   inf_du lg(mu)  ||d||  lg(rg) alpha_du
 alpha_pr  ls
  30 1.3490060e+000 0.00e+000 7.56e-002  -4.4 3.66e-002    -  1.00e
+000 1.51e-001f  1
  31 1.3307865e+000 0.00e+000 6.89e-002  -4.0 3.94e-002    -
 7.81e-001 4.97e-001f  1
  32 1.3193885e+000 0.00e+000 2.22e-001  -3.8 1.21e-002    -
 8.77e-001 1.00e+000f  1
  33 1.3126632e+000 0.00e+000 1.27e-001  -9.8 1.36e-002    -
 7.87e-001 3.78e-001f  1
  34 1.2993745e+000 0.00e+000 1.22e-001  -4.7 2.45e-002    -
 7.16e-001 4.56e-001f  1
  35 1.2922290e+000 0.00e+000 1.23e-001  -5.4 2.75e-002    -
 7.82e-001 2.03e-001f  1
  36 1.2803522e+000 0.00e+000 7.92e-002 -11.0 4.21e-002    -
 3.33e-001 2.09e-001f  1
  37 1.2674119e+000 0.00e+000 9.50e-002  -5.5 6.16e-002    -
 8.71e-001 1.64e-001f  1
  38 1.2622395e+000 0.00e+000 9.00e-002  -5.1 3.41e-002    -
 6.76e-001 1.17e-001f  1
  39 1.2463301e+000 0.00e+000 7.60e-002  -4.9 4.27e-002    -
 7.32e-001 2.61e-001f  1
iter    objective    inf_pr   inf_du lg(mu)  ||d||  lg(rg) alpha_du
 alpha_pr  ls
  40 1.2400471e+000 0.00e+000 1.31e-001  -4.3 1.40e-002    -
 6.61e-001 2.50e-001f  1
  41 1.2325859e+000 0.00e+000 1.16e-001  -4.2 1.88e-002    -
 4.38e-001 2.61e-001f  1
  42 1.2276620e+000 0.00e+000 1.02e-001 -10.3 2.19e-002    -
 5.80e-001 1.47e-001f  1
  43 1.2139467e+000 0.00e+000 1.05e-001  -5.2 3.63e-002    -  1.00e
+000 2.69e-001f  1
```

```
  44 1.2092307e+000 0.00e+000 2.04e-001  -5.7 2.76e-002    -
 7.76e-001 1.18e-001f  1
  45 1.2025930e+000 0.00e+000 1.27e-001  -6.7 2.58e-002    -
 6.64e-001 1.64e-001f  1
  46 1.1918803e+000 0.00e+000 1.49e-001  -5.4 3.25e-002    -  1.00e
+000 2.49e-001f  1
  47 1.1811352e+000 0.00e+000 3.80e-002  -4.5 3.64e-002    -  1.00e
+000 2.92e-001f  1
  48 1.1765723e+000 0.00e+000 8.13e-002  -4.8 2.33e-002    -
 8.55e-001 2.03e-001f  1
  49 1.1704586e+000 0.00e+000 6.06e-002  -5.4 3.19e-002    -  1.00e
+000 2.01e-001f  1
iter    objective    inf_pr   inf_du lg(mu)  ||d||  lg(rg) alpha_du
 alpha_pr  ls
  50 1.1597742e+000 0.00e+000 2.92e-002  -4.6 3.54e-002    -
 4.98e-001 4.10e-001f  1
  51 1.1565150e+000 0.00e+000 1.99e-001  -4.2 1.10e-002    -  1.00e
+000 4.55e-001f  1
  52 1.1532504e+000 0.00e+000 8.96e-002  -6.3 1.78e-002    -
 5.69e-001 1.64e-001f  1
  53 1.1398902e+000 0.00e+000 4.16e-002  -4.8 3.75e-002    -
 5.61e-001 3.97e-001f  1
  54 1.1308042e+000 0.00e+000 2.79e-002  -4.3 3.85e-002    -
 8.22e-001 3.60e-001f  1
  55 1.1223989e+000 0.00e+000 1.25e-001  -4.7 2.04e-002    -
 7.23e-001 5.16e-001f  1
  56 1.2717907e+000 0.00e+000 1.59e-001  -3.4 1.47e-001    -  1.00e
+000 1.00e+000f  1
  57 1.1685623e+000 0.00e+000 9.65e-002  -3.4 4.54e-002    -  1.00e
+000 1.00e+000f  1
  58 1.1578186e+000 0.00e+000 7.16e-002  -3.4 1.25e-002    -  1.00e
+000 1.00e+000f  1
  59 1.1514750e+000 0.00e+000 3.91e-002  -3.4 8.81e-003    -  1.00e
+000 1.00e+000f  1
iter    objective    inf_pr   inf_du lg(mu)  ||d||  lg(rg) alpha_du
 alpha_pr  ls
  60 1.1495486e+000 0.00e+000 6.27e-002  -3.4 1.05e-002    -  1.00e
+000 1.00e+000f  1
  61 1.1480513e+000 0.00e+000 3.68e-002  -3.4 1.09e-002    -  1.00e
+000 1.00e+000f  1
  62 1.1467486e+000 0.00e+000 2.48e-002  -3.4 5.57e-003    -  1.00e
+000 1.00e+000f  1
  63 1.1428323e+000 0.00e+000 4.21e-002  -3.4 1.80e-002    -  1.00e
+000 1.00e+000f  1
  64 1.1390424e+000 0.00e+000 3.00e-002  -3.4 8.28e-003    -  1.00e
+000 1.00e+000f  1
  65 1.1357442e+000 0.00e+000 2.13e-002  -3.4 1.92e-003    -  1.00e
+000 1.00e+000f  1
  66 1.1309438e+000 0.00e+000 1.76e-002  -3.4 5.23e-003    -  1.00e
+000 1.00e+000f  1
  67 1.1269752e+000 0.00e+000 2.58e-002  -3.4 5.71e-003    -  1.00e
+000 1.00e+000f  1
  68 1.1206528e+000 0.00e+000 3.37e-002  -3.4 1.20e-002    -  1.00e
+000 1.00e+000f  1
```

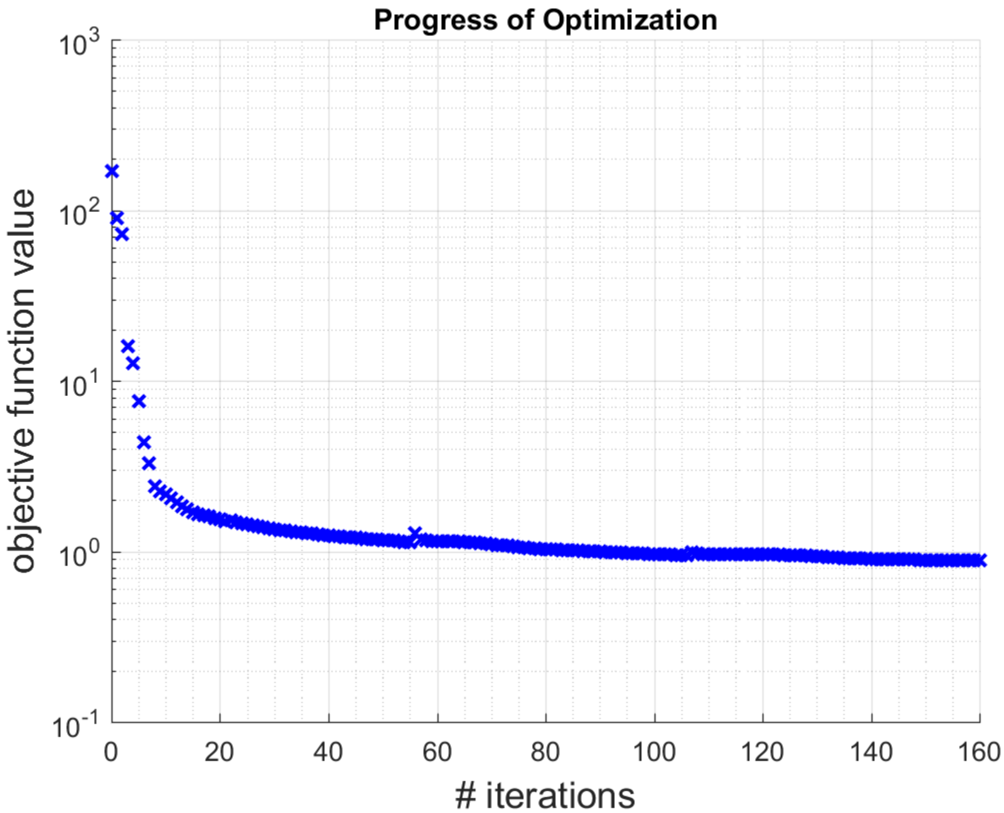
```
  69 1.1120263e+000 0.00e+000 3.22e-002  -4.2 1.07e-002    -  1.00e
+000 1.00e+000f  1
iter    objective    inf_pr   inf_du lg(mu)  ||d||  lg(rg) alpha_du
 alpha_pr  ls
  70 1.1017197e+000 0.00e+000 3.40e-002  -5.6 7.69e-003    -  1.00e
+000 1.00e+000f  1
  71 1.0900689e+000 0.00e+000 2.89e-002  -6.2 1.87e-002    -  1.00e
+000 3.64e-001f  1
  72 1.0848965e+000 0.00e+000 6.03e-002  -6.4 1.52e-002    -  1.00e
+000 1.55e-001f  1
  73 1.0776577e+000 0.00e+000 6.15e-002  -4.0 4.50e-003    -
 9.60e-001 8.37e-001f  1
  74 1.0662697e+000 0.00e+000 1.17e-001  -4.7 1.14e-002    -  1.00e
+000 5.27e-001f  1
  75 1.0570152e+000 0.00e+000 4.08e-002  -5.8 1.34e-002    -  1.00e
+000 4.43e-001f  1
  76 1.0527430e+000 0.00e+000 4.43e-002  -4.1 1.14e-002    -
 9.96e-001 7.81e-001f  1
  77 1.0421556e+000 0.00e+000 1.25e-001  -5.0 1.09e-002    -  1.00e
+000 7.47e-001f  1
  78 1.0396648e+000 0.00e+000 6.81e-002 -10.9 1.23e-002    -
 8.66e-001 1.86e-001f  1
  79 1.0334545e+000 0.00e+000 6.59e-002  -5.2 2.23e-002    -
 7.48e-001 2.97e-001f  1
iter    objective    inf_pr   inf_du lg(mu)  ||d||  lg(rg) alpha_du
 alpha_pr  ls
  80 1.0283782e+000 0.00e+000 8.19e-002  -5.6 2.35e-002    -
 5.66e-001 2.65e-001f  1
  81 1.0266396e+000 0.00e+000 1.23e-001  -7.3 2.37e-002    -
 8.62e-001 9.36e-002f  1
  82 1.0220485e+000 0.00e+000 9.63e-002  -6.7 3.19e-002    -
 9.03e-001 2.00e-001f  1
  83 1.0177696e+000 0.00e+000 6.27e-002  -7.4 3.55e-002    -  1.00e
+000 1.74e-001f  1
  84 1.0154361e+000 0.00e+000 1.05e-001  -5.4 2.34e-002    -
 7.51e-001 1.51e-001f  1
  85 1.0115662e+000 0.00e+000 4.47e-002  -4.8 3.74e-002    -
 7.93e-001 1.73e-001f  1
  86 1.0097005e+000 0.00e+000 7.91e-002  -7.0 2.66e-002    -
 3.71e-001 1.04e-001f  1
  87 1.0062117e+000 0.00e+000 7.76e-002  -4.4 4.86e-002    -
 6.03e-001 1.32e-001f  1
  88 1.0022194e+000 0.00e+000 6.78e-002  -4.3 3.64e-002    -
 5.30e-001 1.89e-001f  1
  89 9.9783179e-001 0.00e+000 8.94e-002 -10.4 3.03e-002    -
 4.30e-001 2.33e-001f  1
iter    objective    inf_pr   inf_du lg(mu)  ||d||  lg(rg) alpha_du
 alpha_pr  ls
  90 9.9526408e-001 0.00e+000 1.80e-001  -5.0 1.93e-002    -
 8.31e-001 2.06e-001f  1
  91 9.9309042e-001 0.00e+000 1.70e-001  -5.5 2.31e-002    -
 6.17e-001 1.32e-001f  1
  92 9.8856582e-001 0.00e+000 8.44e-002  -4.6 2.53e-002    -
 5.15e-001 2.72e-001f  1
```

```
  93 9.8498654e-001 0.00e+000 5.97e-002  -4.9 2.45e-002    -
 4.22e-001 1.95e-001f  1
  94 9.8110315e-001 0.00e+000 4.79e-002 -10.8 3.71e-002    -
 2.77e-001 1.37e-001f  1
  95 9.7828877e-001 0.00e+000 7.91e-002  -4.7 3.19e-002    -
 6.41e-001 1.20e-001f  1
  96 9.7319549e-001 0.00e+000 7.35e-002  -4.8 4.12e-002    -
 4.27e-001 1.74e-001f  1
  97 9.7125917e-001 0.00e+000 7.10e-002 -10.8 4.10e-002    -
 4.44e-001 6.38e-002f  1
  98 9.6717406e-001 0.00e+000 9.47e-002  -5.8 3.02e-002    -
 5.27e-001 1.91e-001f  1
  99 9.6232298e-001 0.00e+000 5.40e-002  -5.2 2.88e-002    -
 6.20e-001 2.69e-001f  1
iter    objective    inf_pr   inf_du lg(mu)  ||d||  lg(rg) alpha_du
 alpha_pr  ls
 100 9.6117072e-001 0.00e+000 7.34e-002  -5.2 7.63e-003    -
 4.30e-001 1.95e-001f  1
 101 9.5866235e-001 0.00e+000 6.79e-002  -4.4 1.38e-002    -
 6.48e-001 3.40e-001f  1
 102 9.5725853e-001 0.00e+000 5.68e-002  -4.6 7.50e-003    -
 4.70e-001 2.90e-001f  1
 103 9.5563347e-001 0.00e+000 6.05e-002 -10.6 1.12e-002    -
 3.15e-001 1.72e-001f  1
 104 9.5348679e-001 0.00e+000 9.81e-002  -5.0 1.77e-002    -
 8.45e-001 1.64e-001f  1
 105 9.4842063e-001 0.00e+000 6.63e-002  -4.6 1.90e-002    -
 6.27e-001 4.98e-001f  1
 106 9.4617285e-001 0.00e+000 4.40e-002  -5.3 2.20e-002    -
 6.76e-001 1.85e-001f  1
 107 9.9103016e-001 0.00e+000 8.35e-002  -3.8 2.16e-002    -
 8.70e-001 1.00e+000f  1
 108 9.7220512e-001 0.00e+000 3.09e-002  -3.9 1.50e-002    -  1.00e
+000 8.38e-001f  1
 109 9.6632343e-001 0.00e+000 2.12e-001  -3.9 7.15e-003    -  1.00e
+000 1.00e+000f  1
iter    objective    inf_pr   inf_du lg(mu)  ||d||  lg(rg) alpha_du
 alpha_pr  ls
 110 9.6287428e-001 0.00e+000 1.42e-002  -3.9 8.17e-003    -
 9.73e-001 1.00e+000f  1
 111 9.6354708e-001 0.00e+000 2.58e-002  -3.9 8.90e-003    -  1.00e
+000 1.00e+000f  1
 112 9.6275569e-001 0.00e+000 1.57e-002  -3.9 3.30e-003    -  1.00e
+000 1.00e+000f  1
 113 9.6411641e-001 0.00e+000 8.58e-003  -3.9 3.09e-003    -  1.00e
+000 1.00e+000f  1
 114 9.6496170e-001 0.00e+000 9.81e-003  -3.9 2.33e-003    -  1.00e
+000 1.00e+000f  1
 115 9.6544646e-001 0.00e+000 1.34e-002  -3.9 4.57e-003    -  1.00e
+000 1.00e+000f  1
 116 9.6395176e-001 0.00e+000 9.50e-003  -3.9 3.21e-003    -  1.00e
+000 1.00e+000f  1
 117 9.6240002e-001 0.00e+000 9.36e-003  -3.9 2.70e-003    -  1.00e
+000 1.00e+000f  1
```

```
 118 9.6022179e-001 0.00e+000 1.17e-002  -3.9 5.04e-003    -  1.00e
+000 1.00e+000f  1
 119 9.5834308e-001 0.00e+000 1.01e-002  -3.9 6.28e-003    -  1.00e
+000 1.00e+000f  1
iter    objective    inf_pr   inf_du lg(mu)  ||d||  lg(rg) alpha_du
 alpha_pr  ls
 120 9.5692798e-001 0.00e+000 1.02e-002  -3.9 6.15e-003    -  1.00e
+000 1.00e+000f  1
 121 9.5631828e-001 0.00e+000 9.43e-003  -3.9 2.57e-003    -  1.00e
+000 1.00e+000f  1
 122 9.5589146e-001 0.00e+000 9.12e-003  -3.9 2.75e-003    -  1.00e
+000 1.00e+000f  1
 123 9.5460755e-001 0.00e+000 1.25e-002  -3.9 4.73e-003    -  1.00e
+000 1.00e+000f  1
 124 9.5356639e-001 0.00e+000 8.97e-003  -3.9 5.00e-003    -  1.00e
+000 1.00e+000f  1
 125 9.5200129e-001 0.00e+000 7.66e-003  -3.9 4.14e-003    -  1.00e
+000 1.00e+000f  1
 126 9.5026339e-001 0.00e+000 8.59e-003  -3.9 4.80e-003    -  1.00e
+000 1.00e+000f  1
 127 9.4736202e-001 0.00e+000 1.45e-002  -3.9 8.28e-003    -  1.00e
+000 1.00e+000f  1
 128 9.4458164e-001 0.00e+000 1.01e-002  -3.9 8.02e-003    -  1.00e
+000 1.00e+000f  1
 129 9.3746848e-001 0.00e+000 1.60e-002  -4.5 1.96e-002    -  1.00e
+000 1.00e+000f  1
iter    objective    inf_pr   inf_du lg(mu)  ||d||  lg(rg) alpha_du
 alpha_pr  ls
 130 9.3042564e-001 0.00e+000 1.71e-002  -5.5 3.35e-002    -  1.00e
+000 4.50e-001f  1
 131 9.2861860e-001 0.00e+000 8.88e-002  -6.1 1.48e-002    -  1.00e
+000 1.31e-001f  1
 132 9.2292080e-001 0.00e+000 3.43e-002 -11.0 1.47e-002    -
 5.63e-001 3.58e-001f  1
 133 9.1881489e-001 0.00e+000 2.61e-002  -5.2 1.03e-002    -
 5.53e-001 2.69e-001f  1
 134 9.1625790e-001 0.00e+000 8.07e-002  -5.8 1.20e-002    -  1.00e
+000 1.81e-001f  1
 135 9.1240944e-001 0.00e+000 4.97e-002  -7.6 1.42e-002    -
 4.05e-001 2.51e-001f  1
 136 9.0989065e-001 0.00e+000 3.01e-002  -5.4 1.35e-002    -
 4.33e-001 1.65e-001f  1
 137 9.0875589e-001 0.00e+000 3.23e-002 -11.0 1.14e-002    -
 4.26e-001 9.94e-002f  1
 138 9.0651262e-001 0.00e+000 3.72e-002  -6.3 1.50e-002    -  1.00e
+000 1.65e-001f  1
 139 9.0420033e-001 0.00e+000 2.63e-002  -5.7 1.48e-002    -
 9.29e-001 1.98e-001f  1
iter    objective    inf_pr   inf_du lg(mu)  ||d||  lg(rg) alpha_du
 alpha_pr  ls
 140 9.0300141e-001 0.00e+000 7.30e-002  -5.8 8.12e-003    -
 9.36e-001 2.08e-001f  1
 141 9.0058145e-001 0.00e+000 5.36e-002  -5.1 1.31e-002    -
 9.96e-001 3.26e-001f  1
```

```
 142 8.9954983e-001 0.00e+000 3.09e-002  -5.4 1.42e-002    -
 6.63e-001 1.23e-001f  1
 143 8.9807712e-001 0.00e+000 5.16e-002  -5.6 9.59e-003    -
 7.45e-001 2.63e-001f  1
 144 8.9639926e-001 0.00e+000 4.59e-002  -5.1 1.13e-002    -
 8.20e-001 3.47e-001f  1
 145 8.9499442e-001 0.00e+000 4.16e-002  -5.9 1.16e-002    -
 5.68e-001 2.23e-001f  1
 146 8.9383728e-001 0.00e+000 2.83e-002  -5.1 1.18e-002    -
 6.35e-001 2.34e-001f  1
 147 8.9237924e-001 0.00e+000 3.19e-002  -5.0 1.39e-002    -
 8.53e-001 4.46e-001f  1
 148 8.9159228e-001 0.00e+000 4.37e-002  -5.3 3.47e-003    -
 9.06e-001 3.70e-001f  1
 149 8.9053598e-001 0.00e+000 3.51e-002 -11.0 6.22e-003    -
 4.21e-001 2.63e-001f  1
iter    objective    inf_pr   inf_du lg(mu)  ||d||  lg(rg) alpha_du
 alpha_pr  ls
 150 8.8968815e-001 0.00e+000 4.40e-002  -6.9 8.64e-003    -
 7.56e-001 1.77e-001f  1
 151 8.8889024e-001 0.00e+000 5.08e-002  -7.1 1.00e-002    -
 9.36e-001 1.58e-001f  1
 152 8.8784494e-001 0.00e+000 5.08e-002  -5.6 8.97e-003    -
 7.42e-001 2.58e-001f  1
 153 8.8938157e-001 0.00e+000 1.26e-001  -4.8 1.86e-002    -
 5.65e-001 9.14e-001f  1
 154 8.8805800e-001 0.00e+000 1.04e-001  -5.0 2.05e-003    -  1.00e
+000 9.84e-001f  1
 155 8.8761731e-001 0.00e+000 5.06e-002  -5.0 3.64e-003    -
 7.33e-001 1.00e+000f  1
 156 8.8708058e-001 0.00e+000 2.76e-002  -5.6 3.26e-003    -
 8.48e-001 2.94e-001f  1
 157 8.8602966e-001 0.00e+000 5.33e-002  -6.1 5.01e-003    -
 8.43e-001 3.43e-001f  1
 158 8.8544657e-001 0.00e+000 5.82e-002  -5.5 4.78e-003    -
 9.74e-001 2.68e-001f  1
 159 8.8471922e-001 0.00e+000 3.82e-002  -6.4 7.62e-003    -
 6.93e-001 2.19e-001f  1
iter    objective    inf_pr   inf_du lg(mu)  ||d||  lg(rg) alpha_du
 alpha_pr  ls
 160 8.8393186e-001 0.00e+000 3.39e-002  -6.6 1.34e-002    -
 9.10e-001 1.61e-001f  1

Number of Iterations....: 160

                                   (scaled)                 (unscaled)
Objective...............:  8.8393185574438515e-001
 8.8393185574438515e-001
Dual infeasibility......:  3.3948503162460439e-002
 3.3948503162460439e-002
Constraint violation....:  0.0000000000000000e+000
 0.0000000000000000e+000
Complementarity.........:  7.8777891974112479e-006
 7.8777891974112479e-006
```

```
Overall NLP error.......:  3.3948503162460439e-002
 3.3948503162460439e-002


Number of objective function evaluations             = 161
Number of objective gradient evaluations             = 161
Number of equality constraint evaluations            = 0
Number of inequality constraint evaluations          = 0
Number of equality constraint Jacobian evaluations   = 0
Number of inequality constraint Jacobian evaluations = 0
Number of Lagrangian Hessian evaluations             = 0
Total CPU secs in IPOPT (w/o function evaluations)   =     17.985
Total CPU secs in NLP function evaluations           =    113.816

EXIT: Solved To Acceptable Level.
Calculating final cubes...
```

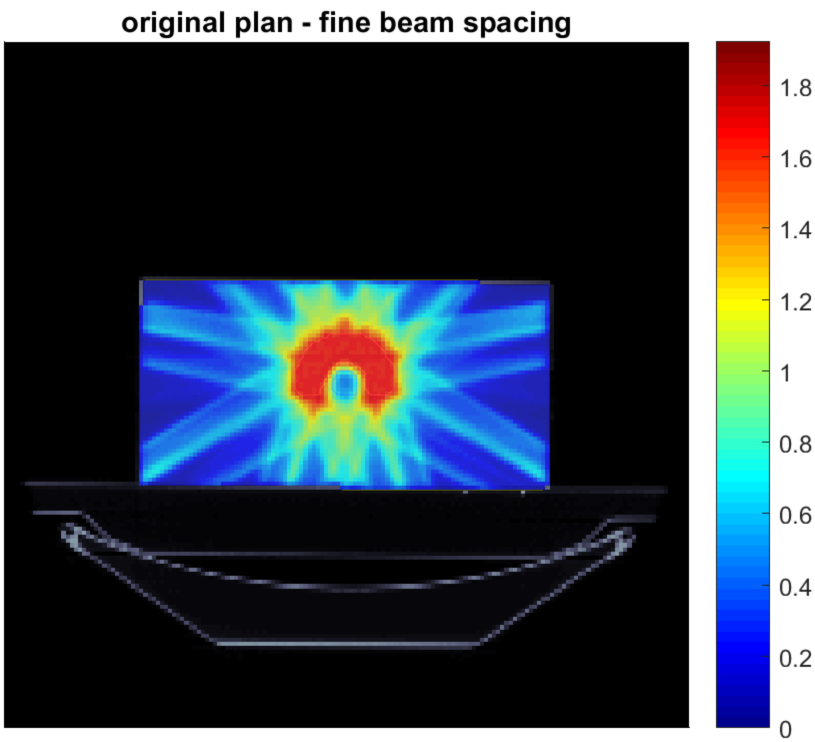


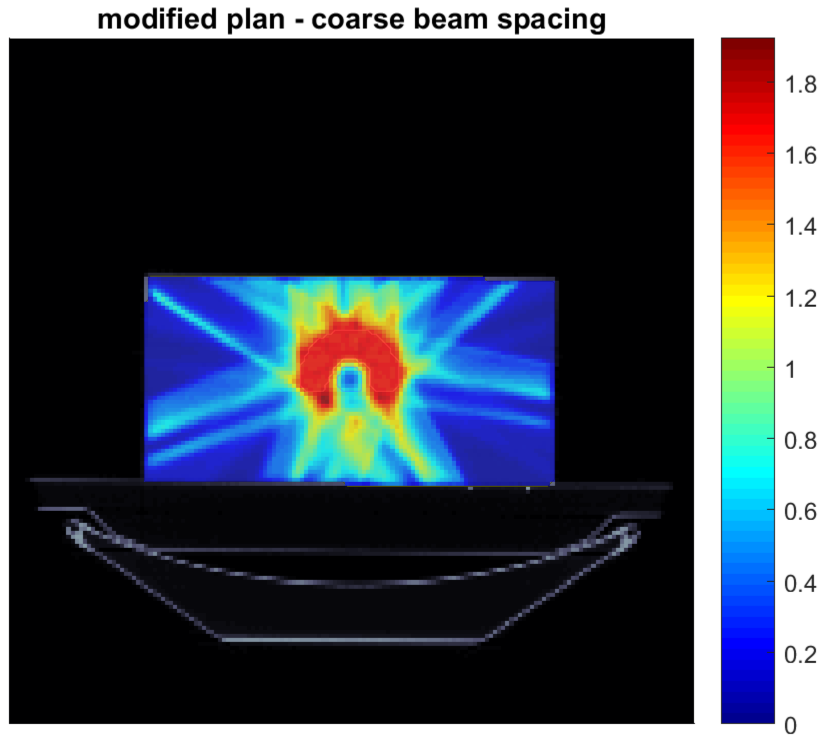
Visual Comparison of results

Let's compare the new recalculation against the optimization result. Check if you have added all subdirectories to the Matlab search path, otherwise it will not find the plotting function

```
plane      = 3;
doseWindow = [0 max([resultGUI.physicalDose(:);
 resultGUI_coarse.physicalDose(:)])];
```

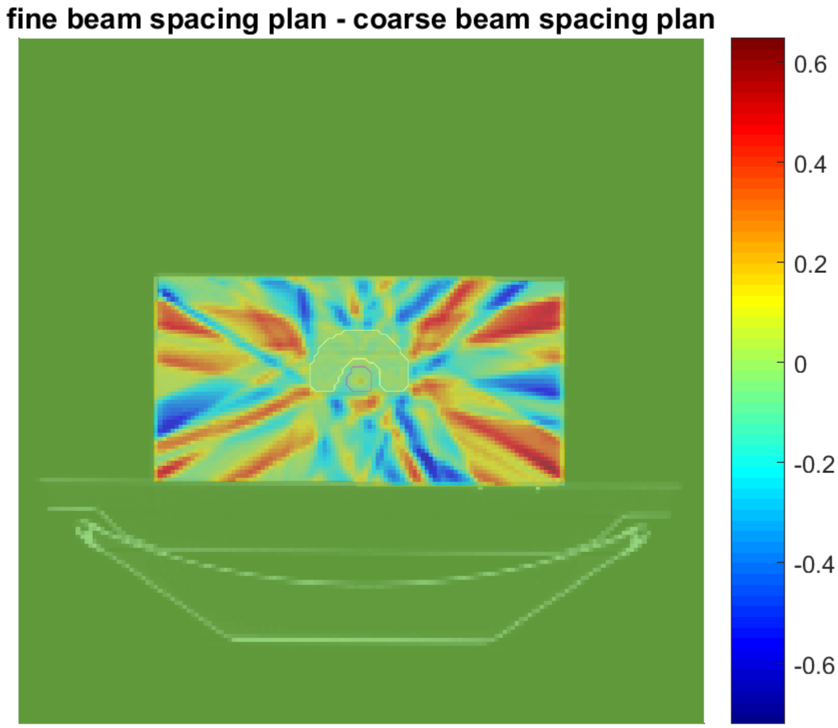
```
figure,title('original plan - fine beam spacing')
matRad_plotSliceWrapper(gca,ct,cst,1,resultGUI.physicalDose,plane,slice,
[],0.75,colorcube,[],doseWindow,[]);
figure,title('modified plan - coarse beam spacing')
matRad_plotSliceWrapper(gca,ct,cst,1,resultGUI_coarse.physicalDose,plane,slice,
[],0.75,colorcube,[],doseWindow,[]);
```





At this point we would like to see the absolute difference of the first optimization (finer beam spacing) and the second optimization (coarser beam spacing)

```
absDiffCube = resultGUI.physicalDose-resultGUI_coarse.physicalDose;
figure,title( 'fine beam spacing plan - coarse beam spacing plan')
matRad_plotSliceWrapper(gca,ct,cst,1,absDiffCube,plane,slice,[],
[],colorcube);
```



Obtain dose statistics

Two more columns will be added to the cst structure depicting the DVH and standard dose statistics such as D95,D98, mean dose, max dose etc.

```
cst        = matRad_indicatorWrapper(cst,pln,resultGUI);
cst_coarse = matRad_indicatorWrapper(cst,pln,resultGUI_coarse);
```

```
  0                  BODY - Mean dose =  0.17 Gy +/-  0.33 Gy (Max dose
 =  1.72 Gy, Min dose =  0.00 Gy)
                             D2% =  1.39 Gy, D5% =  0.90 Gy, D50% =
 0.01 Gy, D95% =  0.00 Gy, D98% =  0.00 Gy,
                             V0Gy = 100.00%, V0.3Gy =  21.01%, V0.6Gy =
  9.66%, V1Gy =   3.91%, V1.3Gy =   2.24%, V1.7Gy =   0.02%,

  1                  Core - Mean dose =  0.58 Gy +/-  0.16 Gy (Max dose
 =  0.87 Gy, Min dose =  0.20 Gy)
                             D2% =  0.87 Gy, D5% =  0.84 Gy, D50% =
 0.55 Gy, D95% =  0.32 Gy, D98% =  0.22 Gy,
                             V0Gy = 100.00%, V0.3Gy =  95.78%, V0.6Gy =
 40.78%, V1Gy =   0.00%, V1.3Gy =   0.00%, V1.7Gy =   0.00%,

  2           OuterTarget - Mean dose =  1.67 Gy +/-  0.02 Gy (Max dose
 =  1.72 Gy, Min dose =  1.60 Gy)
                             D2% =  1.70 Gy, D5% =  1.69 Gy, D50% =
 1.67 Gy, D95% =  1.63 Gy, D98% =  1.62 Gy,
```

```
                          V0Gy = 100.00%, V0.3Gy = 100.00%, V0.6Gy =
100.00%, V1Gy = 100.00%, V1.3Gy = 100.00%, V1.7Gy =   1.18%,
                          CI = 0.8476, HI =  3.50 for reference dose
of 1.7 Gy

 0                BODY - Mean dose =  0.17 Gy +/-  0.34 Gy (Max dose
=  1.92 Gy, Min dose =  0.00 Gy)
                          D2% =  1.45 Gy, D5% =  0.94 Gy, D50% =
0.01 Gy, D95% =  0.00 Gy, D98% =  0.00 Gy,
                          V0Gy = 100.00%, V0.3Gy =  18.90%, V0.7Gy =
 7.93%, V1.1Gy =   3.51%, V1.5Gy =   1.82%, V1.9Gy =   0.00%,

 1                Core - Mean dose =  0.60 Gy +/-  0.17 Gy (Max dose
=  0.90 Gy, Min dose =  0.21 Gy)
                          D2% =  0.89 Gy, D5% =  0.87 Gy, D50% =
0.58 Gy, D95% =  0.36 Gy, D98% =  0.24 Gy,
                          V0Gy = 100.00%, V0.3Gy =  97.50%, V0.7Gy =
33.75%, V1.1Gy =   0.00%, V1.5Gy =   0.00%, V1.9Gy =   0.00%,

 2         OuterTarget - Mean dose =  1.66 Gy +/-  0.02 Gy (Max dose
=  1.72 Gy, Min dose =  1.54 Gy)
                          D2% =  1.70 Gy, D5% =  1.70 Gy, D50% =
1.67 Gy, D95% =  1.63 Gy, D98% =  1.61 Gy,
                          V0Gy = 100.00%, V0.3Gy = 100.00%, V0.7Gy =
100.00%, V1.1Gy = 100.00%, V1.5Gy = 100.00%, V1.9Gy =   0.00%,
                          CI = 0.7568, HI =  3.76 for reference dose
of 1.7 Gy
```

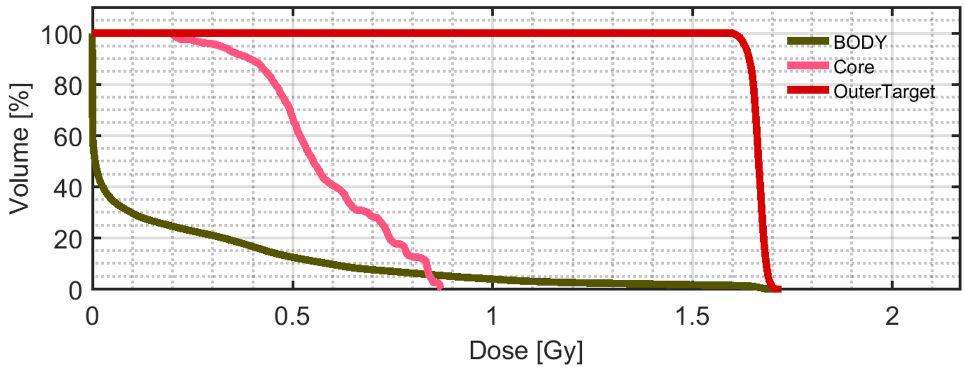
The treatment plan using more beams should in principle result in a better OAR sparing. Therefore lets have a look at the D95 of the OAR of both plans

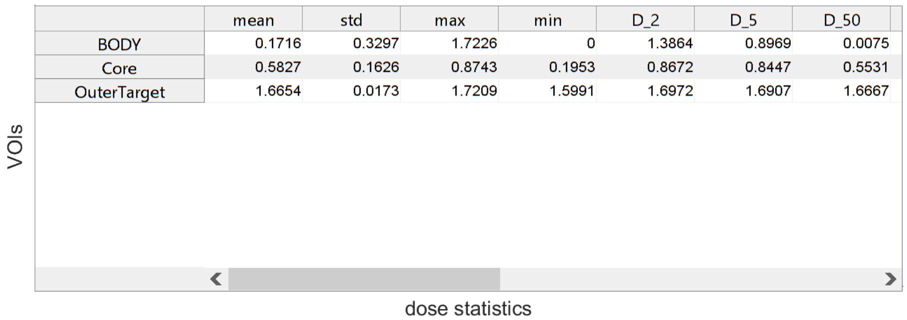
```
ixOAR = 2;
display(cst{ixOAR,9}{1}.D_95);
display(cst_coarse{ixOAR,9}{1}.D_95);
```

```
    0.3213

    0.3626
```

Indeed, the coarse beam plan yields a higher D95 in the OAR. Finally, let's plot the DVH

```
matRad_showDVH(cst,pln);
```





	mean	std	max	min	D_2	D_5	D_50
BODY	0.1716	0.3297	1.7226	0	1.3864	0.8969	0.0075
Core	0.5827	0.1626	0.8743	0.1953	0.8672	0.8447	0.5531
OuterTarget	1.6654	0.0173	1.7209	1.5991	1.6972	1.6907	1.6667

Published with MATLAB® R2016b