Optimization Tools for Radiation Treatment Planning in Matlab

Michael C. Ferris * Jinho Lim † David M. Shepard ‡

May 13, 2003

Abstract

This paper describes a suite of optimization tools for radiation treatment planning within the Matlab programming environment. The data included with these tools was computed for real patient cases using a Monte Carlo dose engine. The formulation of a series of optimization models is described that utilizes this data within a modeling system. Furthermore, visualization techniques are provided that assist in validating the quality of each solution. The versatility and utility of the tools are shown using a sequence of optimization techniques designed to generate a practical solution. These tools and the associated data are available for download from www.cs.wisc.edu/∼ferris/3dcrt.

1 Introduction

The optimization of radiation treatment for cancer has become an active research topic in recent years [2, 3, 4, 7, 13, 14, 28, 31, 32]. Many types of cancer are treated by applying radiation from external sources, firing beams into a patient from a number of different angles in such a way that the targeted tumor lies at the intersection of these beams. The increasing sophistication of treatment devices—the aperture through which the beams pass can take on a variety of shapes, multiples apertures can be delivered

*Computer Sciences Department, 1210 W. Dayton Street, University of Wisconsin, Madison, WI 53706, U.S.A.

†Industrial Engineering Department, 1513 University Ave., University of Wisconsin, Madison, WI 53706, U.S.A.

‡Department of Radiation Oncology, 22 South Greene St., University of Maryland School of Medicine, Baltimore, MD 21201, U.S.A.

for each beam angle, and wedges can be used to vary the radiation intensity across the beam—allows delivery of complex and sophisticated treatment plans, achieving a specified dose to the target area while sparing surrounding tissue and nearby critical structures. Optimization techniques are proving to be useful in the design of such plans.

This paper describes a selection of tools that allow optimization approaches to be applied, visualized and iteratively refined. The tools use the Matlab programming environment for overall control and visualization, and the GAMS modeling language for formulation and solution of the underlying optimization models. One of the strengths of this paper is that we utilize data that corresponds to real patient cases and thus include a variety of inhomogeneities due to different tissue types. Several useful tools for visualization of the results, along with a sophisticated use of an existing interface between the Matlab programming environment and the GAMS modeling language are explained via example. We also show how a succession of optimization problem solutions can be used to satisfy the constraints of a realistic plan, for example dose-volume histogram constraints and the location specific control of hot and cold spots.

The tools described in this paper can be used for creating radiation therapy treatment plans delivered using either of two treatment techniques: (1) three-dimensional conformal radiotherapy (3DCRT) or (2) intensity modulated radiation therapy (IMRT).

3DCRT is the most common delivery technique used in radiation therapy. With this approach, each beam is shaped to match the view of the tumor from the given direction. In addition, one can choose to include a wedge filter in the beam which results in a linear variation in intensity across the beam. This is particularly useful for treating tumors near the patient's surface and for compensating for the curved surface of the patient.

IMRT is a more advanced delivery technique that significantly increases the complexity of radiation delivery but provides an improved ability to conform the radiation to the tumor volume. In IMRT, a nonuniform radiation intensity is delivered from each beam angle. The dose delivered to the tumor volume from each beam angle is typically highly nonuniform, but the total dose delivered to the tumor from all angles provides adequate dose uniformity in the tumor with a rapid falloff in dose in the normal tissue.

While the data and techniques used to solve both types of models have much in common, we restrict the discussion in this paper to 3DCRT for clarity and ease of exposition. Section 2 of the paper describes the data generation for the treatment planning problem. We believe there is a fundamental core of optimization models that are useful for treatment planning.

Several of these optimization models are discussed in Section 3, and the use of the environment in a simple example is given in Section 4. Matlab routines are presented to examine the solution quality in Section 5. All of these routines are available from www.cs.wisc.edu/∼ferris/3dcrt.

We understand that these models should be used in combination and in an iterative fashion to achieve the goals of the planner. Thus, although the core optimization model remains the same, the data that describes a particular instantiation of the model can change in an iterative way as the optimization proceeds. We believe this is where our environment will be most useful. In Section 6, we outline two more complex examples of its use, showing how to incorporate sampling techniques and multiple dvh constraints on a single sensitive structure. While the suite of tools we have developed here are already useful for treatment design and refinement, we believe their strength is their easy extensibility to provide the basis for significant treatment improvements over the coming years.

2 Problem Data

Our planning tool is designed within the Matlab programming environment. The problem data consists of two broad components, namely a set of structures (organs) to which a certain level of radiation must be delivered, and a set of beamlets for delivering this radiation. The amount of data is large and patient/case specific.

We have designed our environment to allow the use of actual patient data as well as simulated data. We have a growing collection of examples of such problems. We have chosen to store these examples as a "gdx" file [29], a "Gams-Data-eXchange" format that allows the data to be accessed very quickly within a GAMS optimization model, and also (via an API) by other programs.

For patient cases, three-dimensional organ geometries are outlined by a physician on a set of CT or MRI images. The physician outlines the GTV ("Gross Tumor Volume," the tumor region) and OARs (for "Organs At Risk," also known as "sensitive structures" or "critical structures"). Since the coordinates of these geometries are continuous, they are not directly usable within our optimization models and have to be converted to a discrete set of voxels. A program "gendata" converts these three-dimensional organ geometries into a set of discrete set of voxel coordinates and stores these (named) sets in the gdx file.

For treatment planning, the Planning Target Volume (PTV) must also

be constructed. To construct the PTV, one begins with the GTV that encompasses known macroscopic disease. Next, a margin is added to include regions of suspected microscopic disease. The new volume is called the clinical target volume (CTV). An additional margin is added to account for anatomical and patient setup uncertainties. The final volume is the PTV. While this procedure is standard, it has some drawbacks in that some voxels may appear in the PTV and also in a sensitive structure. The normal tissue is implicitly stored in the gdx file by saving a rectangular grid of voxels encompassing all the organs of interest.

The second component of the data is the beamlets. The specific details of their generation is given in Section 2.1. We store all the resulting patient specific beamlet data in the same gdx file as the organ structures. It should be noted that the beamlet data is very large and that it does not need to be present in the Matlab environment. We simply use an external program to generate the gdx file, and use the beamlet data only within our optimization models. Such design allows the optimization process to be independent of Matlab if desired. In the 3DCRT case, the beamlet data is given for every voxel indexed over a set of angles and optionally a set of wedges. Section 2.1 describes the generation process in detail, and it can be outlined without continuity loss to the reader.

We also provide a suite of problems based on a water cylinder. These problems can be generated using rotational symmetry from a single beamlet and hence can be stored much more economically. Due to this fact, we ensure our program "gendata" can generate a corresponding gdx file based on this compact representation. The details of the extra input that is needed in this case is given in Appendix A. In the cylinder case, we also provide a Matlab routine "neworgans" to create simulated organ structures, that may be of use in tuning models. This routine enables users to create simulated organ structures within the cylinder.

Finally, the desired or required dose information for each region is specified by the planner, typically as a sequence of dose volume constraints. For example, requirements are of the form "no more than X% of structure A should receive more than Y Gy". For each patient case, these prescriptions are available from the web site.

The organ structures can be manipulated directly within Matlab. In Matlab, each structure is stored as an $m \times 3$ matrix consisting of the m "voxels" that are part of that structure. We provide two routines that allow a user to read a structure from a gdx file and to write a new structure to a gdx file. Access to these structures is useful both for visualization and for sampling as we demonstrate in the sequel.

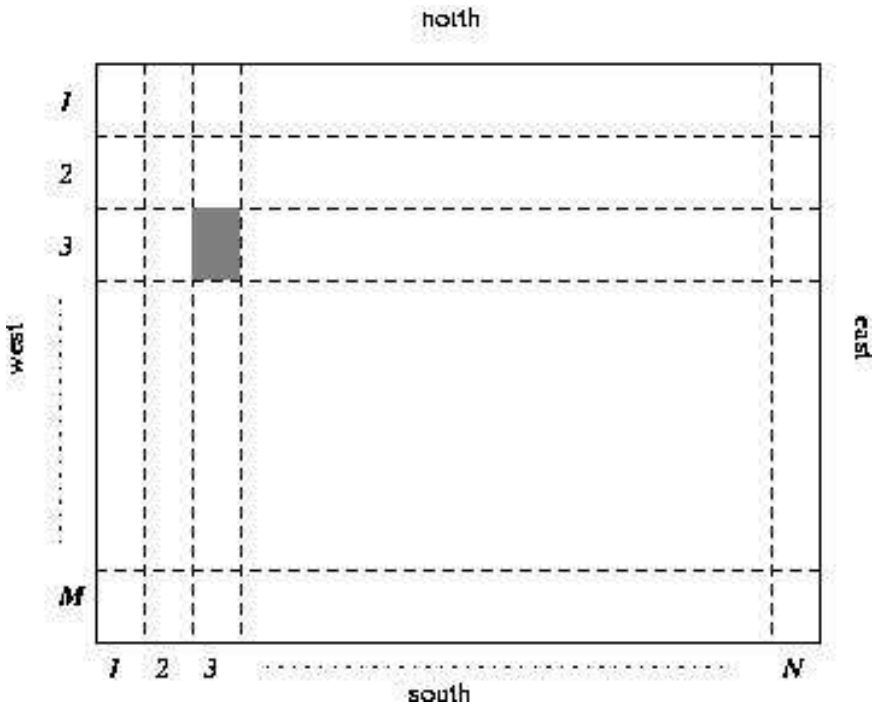

Figure 1: Division of Aperture into Pencil Beams (shaded area represents one beamlet)

2.1 Pencil Beams and Apertures

Modern linear accelerators use a multileaf collimator, located inside the head of the accelerator, to shape the beam of radiation [11, 30]. To calculate the radiation dosage that can be delivered by a beam applied from a given angle, the rectangular aperture obtained by opening the collimator as widely as possible is divided into rectangular subfields arranged in a regular $M \times N$ rectangular pattern, as shown in Figure 1. Each of the subfields is called a *pencil beam* or *beamlet*. M represents the number of leaf pairs in the multileaf collimator, while N represents the number of possible settings we allow for each leaf. We identify each beamlet by the index pair (i, j), where $i = 1, 2, \ldots, M$ and $j = 1, 2, \ldots, N$. In our work, the leaves of the multileaf collimator are 1 cm wide, and a pencil beam is assigned a length of 0.5 cm. Thus, for a 10 cm by 10 cm field, we would use $M = 10$ and $N = 20$, giving a total of 200 beamlets.

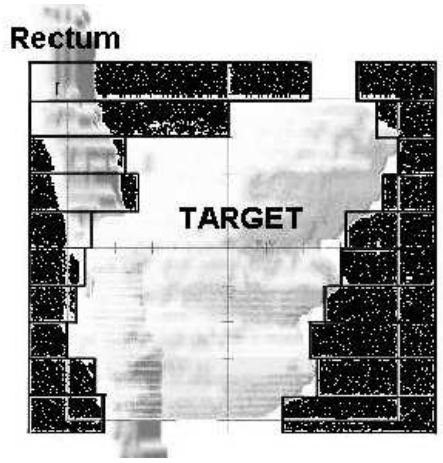

Figure 2: An example of Beam's-eye-view

A separate three-dimensional dose distribution is computed for each pencil beam. The dose distribution matrix for each pencil beam from each angle is calculated using a Monte Carlo technique, which simulates the track of individual radiation particles, for a large number of particles. A unit-intensity, non-wedged beam is assumed for the purposes of these calculations. Each dose distribution consists of the radiation deposited by the beam into each of the small three-dimensional regions ("voxels") into which the treatment area is divided.

As described in Section 1, the pencil beam data sets provided by this tool can be used for either 3DCRT or IMRT. For IMRT optimization, the pencil beam intensities are optimized directly. Thus, an optimized intensity map (fluence map) is produced for each beam angle. In conformal radiotherapy, the shape of each beam is set to match the beam's-eye view (BEV) of the tumor volume, which is essentially the projection of the three-dimensional shape of the tumor onto the plane of the multileaf collimator [5, 6, 8, 11, 19, 21]. One technique for determining the BEV is to employ a ray-tracing algorithm from the radiation source to the tumor volume, setting the beam's-eye view to include all of the rays that pass through the tumor volume. We use an alternative approach based on the dose matrices of the pencil beams. We include in the BEV all pencil beams whose field of significant dose intersects with the target region. To be specific, given a threshold value T, we include a pencil beam in the BEV if its dose delivered to at least one voxel within the target region is at least $T\%$ of the dose delivered by that

pencil beam to *any* voxel. Figure 2 shows an example of a BEV. Once the BEV from a particular angle has been chosen, we can construct the dose matrix for the BEV aperture by simply summing the dose matrices of all the pencil beams that make up the BEV.

The choice of threshold parameter T is critical. If the value of T used in the determining the BEV is too small, the BEV overestimates the target, producing an aperture that irradiates not only the target but also nearby normal tissue and organs at risk. On the other hand, if the value of T is too large, the BEV underestimates the target, and the optimizer might not be able to find a solution that adequately delivers radiation dose within the required range to all parts of the target. The best value of T to use depends somewhat on the shape of the tumor. We choose T as the minimum value such that the resulting BEVs provide a complete 3D coverage of the target from all beam angles considered in the problem. Based on our experiments, a value of T of between 7% and 13% appears to be appropriate.

3 Optimization Models

Optimization models can be used in conjunction with the data outlined above to provide treatment plans that specify how to operate the particular delivery device to generate a dose distribution over the area of interest. The basic optimization models are described more fully in [18], including techniques used to reformulate the problems suitably for optimization solvers.

In conformal radiation therapy, the data that we generate using the procedure outlined above is provided to an optimization model as:

$\mathcal{D}_{(i,j,k),A}$	the dose contribution to voxel (i,j,k) from a beam of weight 1 from angle A,
$\mathcal{T}$	a collection voxels on the target,
$\mathcal{S}$	a collection voxels on the sensitive structure(s),
$\mathcal{N}$	a collection voxels on the normal tissue

When wedges are allowed in the optimization, the data will be provided as

$\mathcal{D}_{(i,j,k),A,F}$	the dose contribution to voxel (i,j,k) from a beam of weight 1 from angle A, using wedge orientation F

3.1 Beam Weight Optimization

The classical optimization problem in conformal radiation therapy is to choose the weights (or intensity levels) to be delivered from a given a set of angles. While much of the literature reformulates the constraints of the problem into the objective using penalization, we propose here to exploit the power of constrained optimization. Thus, these problems can be cast as a quadratic programming problem, minimizing the Euclidean distance between the dose delivered to each voxel and the prescribed dose [7, 22, 26, 27]. Our formulation uses w_A to represent the beam weight delivered from angle A, $D_{(i,j,k)}$ for the total dose deposited to voxel (i, j, k) and λ to represent the relative weighting factors in the objective function. For simplicity we assume that a target prescription of θ is given, the penalty on overdose in the target is the same as for underdose, and that a representative sensitive structure regards dose exceeding ϕ as being hot.

$$\begin{array}{lrcll} \min\limits_{w} & \lambda_t \dfrac{\|D_{\mathcal{T}} - \theta e_{\mathcal{T}}\|_2^2}{\text{card}\,(\mathcal{T})} + \lambda_s \dfrac{\|(D_{\mathcal{S}} - \phi e_{\mathcal{S}})_+\|_2^2}{\text{card}\,(\mathcal{S})} + \lambda_n \dfrac{\|D_{\mathcal{N}}\|_2^2}{\text{card}\,(\mathcal{N})} & & & \\ \text{s.t.} & D_\Omega & = & \sum_{A\in\mathcal{A}} \mathcal{D}_{\Omega,A} w_A, & \Omega = \mathcal{T} \cup \mathcal{S} \cup \mathcal{N}, \\ & l & \leq & D_{\mathcal{T}} \leq u, & \\ & 0 & \leq & w_A, & \forall A \in \mathcal{A}. \end{array} \tag{1}$$

Note that $(\cdot)_+ := max(\cdot, 0)$ can be reformulated using extra constraints and variables as a smooth quadratic as detailed in [18]. Furthermore, we have imposed hard upper and lower bound constraints on the target dose, as well as the objective requirement to be close to the prescription. Other constraints may be useful to deal with prescription constraints at other locations. An implementation of this model is given as *qp.gms* in Appendix C.

Linear programming (LP) has also been extensively used to improve conventional treatment planning techniques [1, 17, 20, 24, 26]. The strength of LP is its ability to control *hot* and *cold* spots or integral dose on the organs using constraints, and the presence of many state-of-the-art LP solvers.

An example of an LP formulation is as follows:

$$\begin{array}{lrcll} \min\limits_{w} & \lambda_t \dfrac{\|(D_{\mathcal{T}} - \theta e_{\mathcal{T}})_+\|_1}{\mathrm{card}\,(\mathcal{T})} + \lambda_s \dfrac{\|(D_{\mathcal{S}} - \phi e_{\mathcal{S}})_+\|_1}{\mathrm{card}\,(\mathcal{S})} + \lambda_n \dfrac{\|D_{\mathcal{N}}\|_1}{\mathrm{card}\,(\mathcal{N})} & & & \\ \text{s.t.} & D_\Omega & = & \sum_{A\in\mathcal{A}} \mathcal{D}_{\Omega,A} w_A, & \Omega = \mathcal{T}\cup\mathcal{S}\cup\mathcal{N}, \\ & l & \leq & D_{\mathcal{T}} \leq u, & \\ & 0 & \leq & w_A, & \forall A \in \mathcal{A}. \end{array} \tag{2}$$

The model here replaces the Euclidean norm objective function with a polyhedral one, for which standard reformulations (see [18]) result in linear programming problems. This model is available as *lp.gms* on the website. While these techniques still suffer from large amounts of data in $\mathcal{D}_{(i,j,k),A}$, they are typically solved in acceptable time frames. These models tend to find optimal solutions more quickly than the corresponding quadratic programming formulations.

Another technique to convert the quadratic (or more generally convex) problem to a linear program is via a piecewise-linear approximation of the objective (see [23]). For a quadratic function, a uniform spacing for the breakpoints guarantees small approximation errors from the piecewise linear interpolant [15]. Since the piecewise linear interpolant is convex, standard techniques can be used to reformulate this as a linear program [12]. The paper [15] suggests a particular formulation and the use of an interior point method to solve the resulting problems.

Recently, some of the medical physics literature has been advocating the use of other forms of objective function in place of the ones outlined above. A popular alternative to those given above is that of generalized equivalent uniform dose (EUD). This is defined on a per structure basis as

$$EUD_a(D,\Omega) := \left(\frac{1}{\mathrm{card}\,(\Omega)} \sum_{(i,j,k)\in\Omega} D^a_{(i,j,k)} \right)^{1/a}.$$

Note that EUD is a scaled version of the a-norm of the dose to the particular structure, and hence is known to be a convex function for any $a \geq 1$ and

concave for $a \leq 1$ [9]. Thus the problem

$$\begin{array}{lll} \max\limits_{w} & & EUD_a(D, \mathcal{T}) \\ \text{s.t.} & D_\Omega = \sum_{A \in \mathcal{A}} \mathcal{D}_{\Omega,A} w_A, & \Omega = \mathcal{T} \cup \mathcal{S} \cup \mathcal{N}, \\ & EUD_b(D, \mathcal{S}) \leq \phi, & \\ & EUD_c(D, \mathcal{N}) \leq u, & \\ & 0 \leq w_A, & \forall A \in \mathcal{A}. \end{array}$$

is a convex optimization problem provided $a \leq 1$ and $b, c \geq 1$. As such, nonlinear programming algorithms will find global solutions to these problems. An example is provided as *eud.gms* on the website.

3.2 Beam angle selection and wedge optimization

Optimization also lends itself to solving the more complex problem of selecting which angles to use as well as their intensities. Mixed Integer Programming (MIP) is a straightforward technique for selecting beams angles among many candidates.

$$\begin{array}{lrcll} \min\limits_{w,\psi} & \lambda_t \dfrac{\|(D_\mathcal{T} - \theta e_\mathcal{T})_+\|_1}{\text{card}\,(\mathcal{T})} + \lambda_s \dfrac{\|(D_\mathcal{S} - \phi e_\mathcal{S})_+\|_1}{\text{card}\,(\mathcal{S})} + \lambda_n \dfrac{\|D_\mathcal{N}\|_1}{\text{card}\,(\mathcal{N})} & & & \\ \text{s.t.} & D_\Omega & = & \sum_{A \in \mathcal{A}} \mathcal{D}_{\Omega,A} w_A, & \Omega = \mathcal{T} \cup \mathcal{S} \cup \mathcal{N}, \\ & l & \leq & D_\mathcal{T} \leq u, & \\ & 0 & \leq & w_A \leq M \psi_A, & \forall A \in \mathcal{A}, \\ & K \geq & \sum\limits_{A \in \mathcal{A}} \psi_A, & & \\ & \psi_A & \in & \{0,1\}, & \forall A \in \mathcal{A}. \end{array} \tag{3}$$

The variable ψ_A is used to determine whether or not to use an angle A for delivery. The choice of M plays a critical role in the speed of the optimization; further advice on its choice is given in [18]. This is implemented as *optangle.gms*.

Finally, we describe an optimization model that simultaneously optimizes beam angles, wedge orientations, and beam intensities. Wedges are placed in front of the collimator to produce a gradient over the dose distribution and can be effective for reducing dose to organs at risk. This can be

done by adding an extra dimension F to the variable w_A:

$$\begin{array}{lrcll} \min\limits_{w,\psi} & \lambda_t \dfrac{\|(D_{\mathcal{T}} - \theta e_{\mathcal{T}})_+\|_1}{\text{card}\,(\mathcal{T})} + \lambda_s \dfrac{\|(D_{\mathcal{S}} - \phi e_{\mathcal{S}})_+\|_1}{\text{card}\,(\mathcal{S})} + \lambda_n \dfrac{\|D_{\mathcal{N}}\|_1}{\text{card}\,(\mathcal{N})} & & & \\ \text{s.t.} & D_\Omega & = & \sum_{A\in\mathcal{A}} \mathcal{D}_{\Omega,A,F} w_{A,F}, & \Omega = \mathcal{T}\cup\mathcal{S}\cup\mathcal{N}, \\ & w_{A,F} & \leq & M\cdot\psi_A, & \\ & l & \leq & D_{\mathcal{T}} \leq u, & \\ & K \geq & \sum\limits_{A\in\mathcal{A}} \psi_A, & & \\ & \psi_A & \in & \{0,1\}, & \forall A\in\mathcal{A}. \end{array} \tag{4}$$

Note that the data for this problem is considerably larger, increasing by a factor related to the number of wedge orientations allowed. An implementation of (4) is available as *optwedge.gms*. Additional optimization models relating to (4) can also be found in [18].

It should be noted that beam energy and non-coplanar beams can also be treated in a similar fashion to wedges within this framework at the cost of increasing the amount of data.

The models described above have been implemented within the GAMS modeling system. Appendix C gives an example for model (1). Most of the notation used in this GAMS file tries to imitate the mathematical symbols used in (1) with a few exceptions: PTV represents $\mathcal{T}$, OAR is for $\mathcal{S}$, Normal is used for $\mathcal{N}$, and sumDose represents D. For debugging purposes, this model can be executed directly at the command prompt:

```
% gams qp
```

Note that "matsol.gms" is a reserved file for the system.

4 Optimization Process

We demonstrate the entire treatment planning process using a set of prostate data that is generated by our phantom cylinder configuration.. There are two organs in this example, namely "prostate" and "rectum". The tumor volume has 5245 voxels, while the rectum consists of 1936 voxels. We are interested in optimizing beam intensities of 36 beam angles for a treatment plan.

We take two steps to generate data that is needed for the rest of treatment planning. The first step is to collect appropriate input data for the

Matlab command "gendata" using the second example given in Appendix A. It creates a Matlab structure array "prob" with data values necessary to utilize the "prostate" example. The second step is to produce the necessary data for the optimization using the inputs (prob) created above:

```
gendata(prob);
```

This generates both a GAMS include file (initdata.gms) and a GDX (GAMS Data Exchange) file [29], typically named *data.gdx*, that are used in the GAMS models. The file *initdata.gms* is a problem specific file that defines sets and parameters that are used in the optimization model. It is described in more detail in Appendix B. It is included at the very beginning of any GAMS files in our toolbox:

```
$include initdata.gms;
```

A Matlab program "rungms" generates a treatment plan based on this data:

```
Dose = rungms('qp');
```

This command uses the interface [10] to execute the GAMS program "qp.gms" of Appendix C. At the end of the run, the Matlab variable "Dose" contains the dose that is delivered as determined by the optimization of the model (1). If necessary, GAMS options can be added followed immediately after the GAMS file name. The last two lines of "qp.gms" facilitate the return of the Dose variable to the Matlab environment.

Since it is typical that a user will wish to update the various organs that are contained in the model and visualize the DVH plots of various structures in the problem, we provide a Matlab command "readstruct" to retrieve the coordinates from the GDX file. For example,

```
PTV = readstruct('data.gdx','prostate');
```

retrieves the three-dimensional coordinates of "prostate" from "data.gdx" into PTV. These coordinates are stored in a Matlab matrix that has a positive number of rows and 3 columns. Each row holds the three dimensional coordinates of a given voxel in the structure.

More complex use of "rungms" is as follows.

```
rungms('GAMS_file_name [-options]',organ1,organ2,...,data);
```

Users can specify the GAMS solver options immediately after the GAMS file name. The additional (optional) input arguments are Matlab structures representing organs. Each organ structure must have a name field. The

name field must be the string that is the set name used in GAMS. For an example, we define a Matlab structure array for the target, another for the sensitive structure, and the third for the normal tissue as follows:

```
target    = struct('name','prostate');
sensitive = struct('name','rectum','data',samplerect);
normal    = struct('name','Normal');
```

In the above example, the target (prostate) is extracted from the "data.gdx" file, whereas the sensitive structure (rectum) uses the voxels provided in the samplerect matrix (which has the same format as that described above for PTV). The M-file "rungms" automatically writes out "samplerect" to a GDX file (rectum.gdx) using a Matlab command "writestruct":

```
writestruct('rectum.gdx',samplerect);
```

The set normal contains the set of those voxels that are neither target nor sensitive and is generated automatically.

The specified GAMS file is executed and returns the three-dimensional matrix of the final dose distribution. This can be used to evaluate the treatment quality using DVH and dose distribution plots as will be explained in the next section.

A final output is always returned, namely the intensities w:

```
[Dose,w] = rungms('qp',target);
```

In the "rungms" example immediately above, the intensities are returned in w. Note that the "qp" model will only have a target structure in this case, i.e. the sensitive and normal structures will be empty in the optimization problem.

The amount of voxels in normal can be unnecessarily large. We provide a Matlab command "genrind" that can be useful to reduce the amount of normal voxels. For example,

```
genrind(PTV,5);
```

generates a set of three-dimensional voxel coordinates that surrounds the PTV within a distance of five voxels. The rind of the PTV can be used as a substitute for the normal structure to reduce the number of voxels in the optimization.

Thus the following Matlab commands set up a rind of the PTV as normal tissue, again with no sensitive structures included in the optimization:

```
normal = struct('name','Normal','data',genrind(PTV,5));
Dose = rungms('qp',target,normal);
```

While "initdata.gms" contains default values that allow the model to be run, we also provide a mechanism to update the values of the parameters in the optimization model. For example,

```
data = struct('kBeams',6,'thetaL',0.97,'thetaU',1.05,...
         'phi',struct('rectum',0.4),...
         'lambda',struct('prostate',2,'normal',0.5));
Dose = rungms('qp',target,sensitive,normal,data);
```

generates new values for various parameters in "qp.gms". The data argument (a Matlab structure) must be the last argument to "rungms". To distinguish it from the organ structures, it cannot have a 'name' field. The remaining fields use execution time assignments to update the values of the given fields. Thus, $kBeams$ gets reset to the value 6, θ_L to the value 0.97, etc. For parameters defined over one dimensional sets (e.g. ϕ), we update a subset of its components by specifying new values in a structure. Thus ϕ_{rectum} will be reset to 0.4 in the above example. The communication mechanism to facilitate this is a file "updatedata.gms".

5 Visualization

Medical experts rely on visualizations of the dose to examine the quality of treatment plans, instead of the objective function that operations researchers typically employ. Two such visualizations are the dose volume histogram (DVH) and a slicewise dose distribution plot. The Matlab routines "dvh" and "doseplot" are provided to allow both sets of users to determine if the quality of solutions are in broad agreement.

5.1 Plotting DVH

The quality of a treatment plan is typically specified and evaluated using a DVH. The DVH shows what fraction of the volume receives at least a certain level of dose. To make a DVH plot of the current solution, the final dose distribution and sets of three-dimensional organ coordinates are passed to a Matlab routine "dvh". "Dose" must be the first input argument for "dvh", followed by the organs of interest:

```
dvh(Dose,PTV,OAR);
```

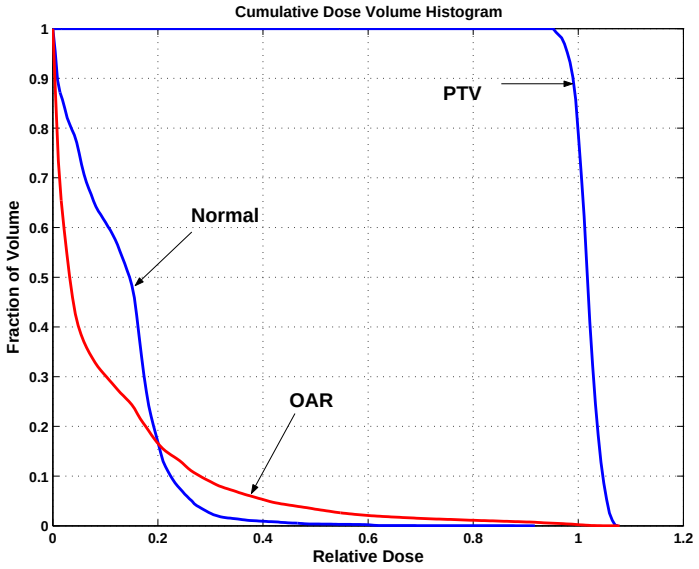

Figure 3: Dose Volume Histogram

This generates a Matlab figure with dose volume histograms for the specified organs. Optionally, the user can specify a line property for the DVH plot as follows:

```
dvh(Dose,PTV,'b-',OAR,'r');
```

where the color blue ('b-') with a solid line is specified for the "PTV" and red ('r') for the "OAR."

An example DVH is shown in Figure 3. The *X-axis* is normalized so that the target prescribed dose (θ) is one. The *Y-axis* represents the fraction of the volume. For example, the line of the normal tissue approximately passes through the coordinate $(0.2, 0.2)$. This means that 80% of the normal tissue receives 20% or less of the target prescribed dose level. Note that the labels on the structures are created manually using the Matlab figure editor; alternatively the "legend" command could be used.

5.2 Plotting dose distribution

Although a DVH provides information about the fraction of each organ receiving each dose level, it does not give spatial information with regards to the location of "hot spots", "cold spots", or "streaking". Visualizing the

dose distribution becomes important to finalize treatment plans for practical use.

To meet this need, a Matlab routine "doseplot" is provided to examine the dose distribution from different viewpoints: axial, sagittal, and coronal. To visualize the dose distribution of the current solution from the axial viewpoint, the following command suffices:

```
doseplot(Dose);
```

This produces a Matlab figure with dose distribution of the current solution using the default viewpoint ("axial" slice). All slices can be viewed one slice at a time by pressing any button on the keyboard. Optional arguments are also allowed. These include sets of three-dimensional organ coordinates, a string ("axial", "coronal" or "sagittal") representing the viewpoint of the image, and a vector of slice numbers:

```
doseplot(Dose,PTV,OAR,'axial');
doseplot(Dose,PTV,OAR,'axial',15:20);
```

In the above examples, the PTV and OAR structures will be outlined on the same slice of the dose distribution. The slice number must come after the choice of the viewpoint. The PTV and OAR structures are indicated on the resulting "axial" output as shown in Figure 4. The second case allows only slices 15 to 20 to be displayed as opposed to all slices. Note that the color blue represents the cold spots while red indicates hot spots as depicted in the Matlab color bar located on the right side of Figure 4. For each structure, an optional argument facilitates the change of the outline color of a given organ, e.g.

```
doseplot(Dose,PTV,'b',OAR,'coronal');
```

outlines the target in blue on all coronal slices.

6 Example usage of toolbox

We outline the use of some of the tools described above for a particular example of 3D conformal radiation therapy treatment planning in a prostate case. We use the model format (3) that has been encoded in a GAMS file "optangle.gms".

To start the process, we use the Matlab command "gendata" to generate files *initdata.gms* and *data.gdx* as discussed in Section 4. These files are required for running optimization models. Using the first example of Appendix A,

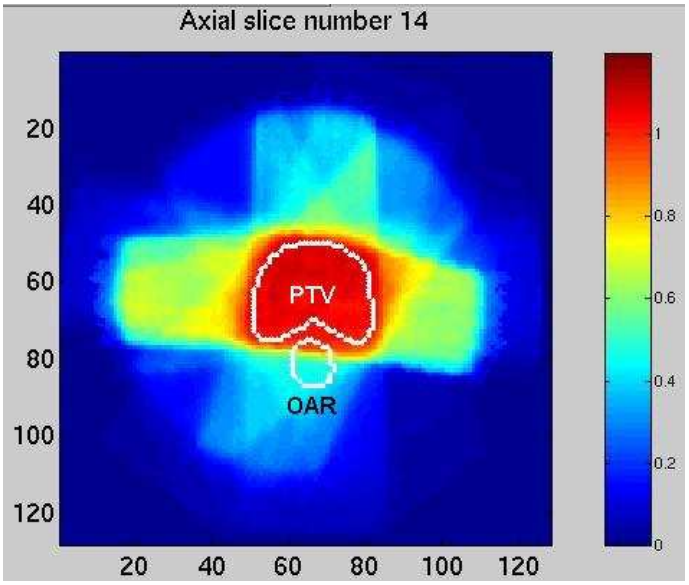

Figure 4: Dose distribution plot in axial slice

```
gendata(prob);
```

generates "initdata.gms" and uses the existing "data.gdx" file. Alternatively, a new GDX file can be generated as discussed in the second example of Appendix A.

A Matlab command "readstruct" is used to extract the organ geometries from the GDX file into the Matlab workspace:

```
prostate = readstruct('data.gdx','prostate');
rectum   = readstruct('data.gdx','rectum');
```

As discussed in Section 4, some parameters of the optimization model can be updated as follows:

```
data = struct('kBeams',6,'thetaL',0.97,'thetaU',1.05,...
         'phi',struct('rectum',0.4),...
         'lambda',struct('normal',0.5));
```

6.1 Data Sampling

A large number of voxels comprise the normal tissue. Although voxels in the normal tissue are important for the final treatment plan, it is shown that

using sampled voxels of the normal structure in the optimization problem has an insignificant effect on the optimal treatment plan dose distribution [18]. A random sampling of voxels can also be used to speed up the computation [25].

In addition to these data reduction techniques, rather than using the complete sets of organ structures, we follow an interactive approach that tries to determine promising beam angles based on an $\alpha\%$ sampling of the sensitive structure as outlined in [18]. The sampling scheme is also noted elsewhere [16].

A self-explanatory example of our sampling scheme is as follows:

```
goodAngles = zeros(prob.nAngle,1);
rate = 0.1;     nsamples = 10;
for s = 1:nsamples
    % sample a subset of prostate
  sample = find(rand(length(prostate),1) < rate);
  target = struct('name','prostate','data',prostate(sample,:));
    % sample a subset of rectum
  sample = find(rand(length(rectum),1) < rate);
  sens   = struct('name','rectum','data',rectum(sample,:));
    % sample a subset of normal
  sample = find(rand(length(normal),1) < rate/10);
  snormal   = struct('name','normal','data',normal(sample,:));
    % run a GAMS optimization model
  [Dose,w]  = rungms('optangle lo=1 optfile=1 optcr=0.03',...
      target,sens,snormal,data);
  used   = find(w > 0);
  goodAngles(used) = goodAngles(used) + 1;
end
```

We used these sampled problems (that solve very quickly) to determine which subset of the angles are promising. The Matlab vector "goodAngles" contains a count of sample how many solutions used as given angle. We then resolve the full model using just those angles that were used in more than one sample case..

```
fixangles = find(goodAngles<=1)-1;  % index adjustment
wup    = [fixangles,zeros(length(fixangles),1)];
data.wup  = wup;
```

Note that the Matlab structure "data" holds new values for parameters that are already declared in the GAMS model "optangle". In particular,

the parameter "wup" is used to set new upper bounds on the weights in the model. By resetting some of these to be zero, the model ignores the corresponding bad angles.

We continue to sample the normal structure, but include all those voxels that are close to the target.

```
target = struct('name','prostate');
sens   = struct('name','rectum');
sample = find(rand(length(normal),1) < rate);
newnormal = [genrind(prostate,3);normal(sample,:)];
snormal   = struct('name','normal','data',newnormal);
[Dose,w]  = rungms('optangle lo=1 optfile=1 optcr=0.03',...
    target,sens,snormal,data);
dvh(Dose,prostate,rectum);
```

6.2 Refining solutions

At this stage, the dvh may show a sensitive structure that is violating a "dvh" constraint. To rectify this situation, we have three recourses. We can update the parameter ϕ in (3) using:

```
data.phi.rectum = 0.2;
```

This will penalize dose on the rectum that exceeds 0.2, instead of the original value of 0.4. Secondly, we can update the penalty parameters in relation to the organ structures:

```
data.lambda = struct('prostate',0.9,'rectum',2,'normal',0.2);
```

Thirdly, we can update the structure itself to allow a subset of the problem voxels to violate the constraints:

```
rectumBottom = extract(Dose,rectum,'bottom',0.82);
```

By executing the above line, the Matlab command "extract" sorts the values of the dose delivered to all voxels in the rectum. Then it exracts (the bottom) 82% of the voxels that received the low dose. The sensitive structure is updated using this new set of voxels in the rectum:

```
sensitive = struct('name','rectum','data',rectumBottom);
```

This is a heuristic, based on a dvh that allows 20% of the voxels to be essentially ignored, we actually ignore slightly less than this. Note that the command "extract",

```
extract(dose_matrix,organ_coordinates,option_string,value);
```

offers two different option types for user to sample a faction of voxels in the specified organ structure: *absolute option* and *relative option*. The relative option was displayed in the previous example of "extract". The same command line can be replaced by

```
rectumBottom = extract(Dose,rectum,'top',0.18);
```

Two options are available for the absolute option: "above" and "below".

```
extract(Dose,rectum,'above',0.9);
```

can be used to extract all voxels that received more than 90% of the target prescribed dose in the rectum.

Resolving the newly updated problem,

```
Dose = rungms('optangle',target,sensitive,data);
dvh(Dose,prostate,rectum);
```

gives a new dvh. Furthermore, other changes to the parameters of the formulation may be carried out, or changes to the formulation itself can be implemented by simply updating the GAMS model file, or the Matlab solution process.

Another example where iterative solution can be useful is in the treatment of location specific hot or cold spots. The dvh plot does not distinguish between the locations of such spots, but clinically cold spots in the center of a tumor location are more serious than those on the periphery. If one of the models (1), (2) or (3) is used, then this should not be a problem since the model incorporates hard upper and lower bound constraints on the dose in the tumor. However, in practice, these hard constraints can lead to infeasibilities or overly homogeneous solutions, and may be dropped or relaxed within the modeling process.

To isolate voxels on the periphery of a given structure we can use the routine "genrind". In the following example, we extract the "internal" rind of size 3 from the tumor (in this case, the prostate):

```
genrind(prostate,-3);
```

To find the cold spots that are centrally located in the tumor we invoke the routine "extract" to find the voxels that are dosed below a given cutoff value, and throw away those voxels that are close to the periphery as follows:

```
coldVox = extract(Dose,prostate,'below',0.9);
coldVox = setdiff(coldVox,genrind(prostate,-3),'rows');
```

Similar techniques could be used to find hot spots within a sensitive structure.

In order to pass new constraints on these substructures to the optimization routines, we use any number of additional organ structures that our toolbox provides by default. These additional organs are called $org1, \cdots, org100$. Thus, to impose a strict lower bound of 0.95 on the dose in the center of the tumor, the following additional lines would suffice:

```
coldTarg = struct('name','org1','data',coldVox);
data.DoseLo.org1 = 0.95;
Dose = rungms('optangle',target,sensitive,coldTarg,data);
```

The additional organs could also be subject to soft constraints - the user just has to define ϕ and λ appropriately. For example,

```
data.lambda.org1 = 10;
Dose = rungms('optangle',target,sensitive,coldTarg,data);
```

Streaking control: When relatively few angles are used for treatment, "streaking" can occur in the normal tissue. Essentially, high dose is delivered along particular rays that due to the decay mechanism results in hot spots of radiation close to beam entry locations. These hot spots must be removed before the treatment plan is finalized. We take two steps to do this. First the normal structure is divided into two distinct sets *innormal* and *outnormal*. The set "innormal" is an external rind of the target structure while "outnormal" is defined as the remainder. Second, a strict dose upperbound is applied only on "outnormal". Note that "innormal" is not considered in the optimization model. The following lines of Matlab code shows an implementation of this method that also uses sampling to reduce the normal voxels considered.

```
sample    = find(rand(length(normal),1) < rate/10);
innormal  = genrind(prostate,6);
outnormal = setdiff(normal(sample,:),innormal,'rows');
snormal   = struct('name','normal','data',outnormal);
data.DoseUp.normal = 0.8;
[Dose,w]  = rungms('optangle lo=1 optfile=1 optcr=0.03',...
    target,sens,snormal,data);
```

The same technique using additional organs $org1, \cdots, org100$ could be used to impose different bounds on different pieces of the structure. Thus to remove hot spots in the normal tissue close to the tumor whilst maintaining streaking control, we would use an additional organ $org2$, for example:

```
org2   = struct('name','org2','data',innormal);
data.DoseUp.org2 = 1.05;
[Dose,w]  = rungms('optangle lo=1 optfile=1 optcr=0.03',...
    target,sens,snormal,org2,data);
```

7 Conclusions

We have provided clinical data for radiation therapy treatment planning within the Matlab environment. We have shown how to use a small suite of programs (gendata, rungms, genrind, extract, dvh, doseplot) in conjunction with a library of optimization model to provide an effective and adaptive treatment planning procedeure. We have demonstrated the utility of the Matlab environment to facilitate more complex data and optimization control.

References

[1] G.K. Bahr, J.G. Kereiakes, H. Horwitz, R. Finney, J. Galvin, and K. Goode. The method of linear programming applied to radiation treatment planning. *Radiology*, 91:686–693, 1968.

[2] T. Bortfeld and W. Schlegel. Optimization of beam orientations in radiation-therapy - some theoretical considerations. *Physics in Medicine and Biology*, 38(2):291–304, 1993.

[3] Th. Bortfeld, J. Burkelbach, R. Boesecke, and W. Schlegel. Methods of image reconstruction from projections applied to conformation radiotherapy. *Physics in Medicine and Biology*, 25(10):1423–1434, 1990.

[4] Thomas Bortfeld, Arthur L. Boyer, Wolfgang Schlegel, Darren L. Kahler, and Timothy J. Waldron. Realization and verification of three-dimensional conformal radiotherapy with modulated fields. *International journal of radiation oncology: Biology, Physics*, 30(4):899–908, 1994.

[5] L. Brewster, G. S. Mageras, and R. Mohan. Automatic-generation of beam apertures. *Medical Physics*, 20(5):1337–1342, 1993.

[6] G.T.Y. Chen, D. R. Spelbring, C.A. Pelizzari, J.M. Balter, L. C. Myrianthopoulous, S. Vijayakumar, and H. Halpern. The use of beam eye view volumetrics in the selection of noncoplanar radiation portals. *International Journal of Radiation Oncology: Biology, Physics*, 23:153–163, 1992.

[7] Yan Chen, Darek Michalski, Christopher Houser, and James M Galvin. A deterministic iterative least-squares algorithm for beam weight optimization in conformal radiotherapy. *Physics in Medicine and Biology.*, 47:1647–1658, 2002.

[8] B. C. J. Cho, W. H. Roa, D. Robinson, and B. Murray. The development of target-eye-view maps for selection of coplanar or noncoplanr beams in conformal radiotherapy treatment planning. *Medical Physics*, 26(11):2367–2372, 1999.

[9] Beong Choi and Joseph O Deasy. The generalized equivalent uniform dose function as a basis for intensity-modulated treatment planning. *Physics in Medicine and Biology*, 47:3579–3589, 2002.

[10] Michael C. Ferris. MATLAB and GAMS interfacing optimization and visualization soft ware. Technical Report Mathematical Programming Technical Report 98-19, Computer Sciences Department, University of Wisconsin, Madison, Wisconsin, 1998.

[11] Michael Goitein, Mark Abrams, Serek Rowell, Helen Pollari, and Judy Wiles. Multi-dimensional treatment planning: Ii. beam's eye-view, back projection, and projection through ct sections. *International Journal of Radiation Oncology: Biology, Physics*, 9:789–797, 1983.

[12] J.K. Ho. Relationships among linear formulations of spearable convex piecewise linear programs. *Mathematical Programming Study*, 24:126–140, 1985.

[13] Intensity Modulated Radiation Therapy Collaborative Working Group. Intensity-modulated radiotherapy: Current status and issues of interest. *International Journal of Radiation Oncology: Biology, Physics*, 51(4):880–914, 2001.

[14] Thomas J. Jordan and Peter C. Williams. The design and performance characteristics of a multileaf collimator. *Physics in Medicine and Biology.*, 39:231–251, 1994.

[15] S Kontogiorgis. Practical piecewise-linear approximation for monotropic optimization. *Informs Journal on Computing*, 12(4):324–340, 2000.

[16] M. Langer, R. Brown, M. Urie, J. Leong, M. Stracher, and J. Shapiro. Large-scale optimization of beam-weights under dose-volume restrictions. *International Journal of Radiation Oncology: Biology, Physics*, 18:887–893, 1990.

[17] M. Langer and J. Leong. Optimization of beam weights under dose-volume restriction. *International Journal of Radiation Oncology: Biology, Physics*, 13:1255–1260, 1987.

[18] Jinho Lim, Michael C. Ferris, Stephen J. Wright, David M. Shepard, and Matthew A. Earl. An optimization framework for conformation radiation treatment planning. *Optimization Technical Report 02-08, Computer Sciences Department, University of Wisconsin - Madison*, December 2002.

[19] D. L. McShan, B. A. Fraass, and A. S. Lichter. Full integration of the beam's eye view concept into computerized treatment planning. *International Journal of Radiation Oncology: Biology, Physics*, 18:1485–1494, 1989.

[20] S. Morrill, R. Lane, J. Wong, and I. I. Rosen. Dose-volume considerations with linear programming. *Medical Physics*, 6(18):1201–1210, 1991.

[21] L.C. Myrianthopoulos, G.T.Y. Chen, S. Vijayakumar, H. Halpern, D. R. Spelbring, and C.A. Pelizzari. Beams eye view volumetrics - an aid in rapid treatment plan development and evaluation. *International Journal of Radiation Oncology: Biology, Physics*, 23(367-375), 1992.

[22] A. T. Redpath, B. L. Vickery, and D. H. Wright. A new technique for radiotherapy planning using quadratic programming. *Physics in Medicine and Biology*, 21:781–791, 1976.

[23] H. E. Romeijn, R. K. Ahuja, and J. F. Dempsey. A new linear programming approach to radiation therapy treatment planning problems. Technical report, University of Florida, 2003.

[24] I. I. Rosen, R. Lane, S. Morrill, and J. Belli. Treatment plan optimization using linear programming. *Medical Physics*, 18(2):141–152, 1990.

[25] C.G. Rowbottom, V.S. Khoo, and S. Webb. Simultaneous optimization of beam orientations and beam weights in conformal radiotherapy. *Medical Physics*, 28(8):1696–1702, 2001.

[26] D. M. Shepard, M. C. Ferris, G. Olivera, and T. R. Mackie. Optimizing the delivery of radiation to cancer patients. *SIAM Review*, 41:721–744, 1999.

[27] G. Starkschall. A constrained least-squares optimization method for external beam radiation therapy treatment planning. *Medical Physics*, 11:659–665, 1984.

[28] J. Tervo and P. Kolmonen. A model for the control of a multileaf collimator in radiation therapy treatment planning. *Inverse Problems*, 16:1875–1895, 2000.

[29] Paul van der Eijk. *GDX facilities in GAMS*. GAMS Contributed Software, http://www.gams.com/contrib/GDXUtils.pdf, 2002.

[30] S. Webb. *The Physics of Conformal Radiotherapy*. Institute of Physics Publishing, 1997.

[31] S. Webb. Configuration options for intensity-modulated radiation therapy using multiple static fields shaped by a multileaf collimator. *Physics in Medicine and Biology.*, 43:241–260, 1998.

[32] Xingen Wu and Yuping Zhu. A global optimization method for three-dimensional conformal radiotherapy treatment planning. *Physics in Medicine and Biology*, 46:109–119, 2001.

A Problem Specification

The routine "gendata" reads in user input that specifies the total number of beam angles considered in the problem, a flag indicating if wedges are to be used, the name of the PTV, GAMS set names for the organ structures,

the GAMS include file name that will store set and parameter definitions, organ geometries, and the dose matrix. This data is provided as a Matlab structure.

For documentation purposes, we have provided Matlab code that generates the appropriate structure for some existing examples. The first example

```
prob = struct(...
  'nAngle',        36,...   % number of beam angles
  'gengdx',        'no',... % generate a GDX file or not
  'use_wedge',     'no',... % use wedge or not
  'PTVname',       'prostate',... % target set name
  'gdxDataName',   'data.gdx',... % GDX data file
  'gdxIncludeFile','initdata.gms',...
  'structures',    {{'prostate','rectum'}});
```

retrieves its data from a GDX file. Note that the flag "gengdx" is set to "no" here since the GDX file is assumed to exist already.

The second example shows how to utilize the existing Beamdata and structure information to generate a new GDX file:

```
prob = struct(...
  'nAngle',        36,...
  'beamcutoff',    9.5,...  % value of T as discussed in Section 2
  'use_wedge',     'no',...
  'gengdx',        'yes',...
  'is_cylinder',   'yes',...
  'baseDir',       '/p/cure-cancer/work1/LIM/',...
  'beamID',        '10x10',...
  'beamDir',       'Beamdata_cylinder',...
  'structDir',     'Structures',...
  'gdxDataName',   'data.gdx',...
  'gdxIncludeFile','initdata.gms',...
  'PTVname',       'prostate',...
  'structures',    struct('prostate','ptv.dat','rectum','rectum.dat'));
```

B initdata.gms

This file is generated automatically from the problem input described in Appendix A. This file has six components. First, basic sets and their dimensions are defined for solving the optimization problems. An example is shown below.

```
sets
        I       /0*125/
        J       /0*125/
        K       /0* 31/
        nAngle  /0*35/
        angle(nAngle);
```

Note that the set "angle" is a subset of all angles considered in the optimization. This set is particularly useful for MIP problems ((3), (4)) to automatically reduce the solution search space. The use of "angle" can be seen in Appendix C.

The next component is to define sets of the three-dimensional coordinates and their dimensions. Sets "prostate" and "rectum" provide coordinates of organs, "PTV", "OAR", and "Normal" are auxiliary set definitions for the target, the sensitive, and the normal structures respectively. The name of the parameter to store the dose distribution is also defined:

```
sets
     prostate(I,J,K), rectum(I,J,K),
     PTV(I,J,K),      OAR(I,J,K),       Normal(I,J,K);
parameter Dose(I,J,K,nAngle);
```

The third component is to define the collection of organ names:

```
set allorgans /'prostate','rectum','Normal', org1*org100/;
```

Note that the set "allorgans" has dummy organs "org1*org100". These extra empty organs play an important role when new organ structures (not defined in the GDX file) are added. As shown in Section 6, an immediate application of this feature can be seen on the local dose control over a subset of an organ structure.

In the next component, global variables for the GAMS model is defined for the target and the critical structures. We then define a set "organs" as the collection of critical structures of interest for the particular instance. In our example, the following three lines

```
$setglobal target   'prostate'
$setglobal critical 'rectum'
set organs(allorgans) /%critical%/;
```

make the global variable target contain "prostate" and the variable critical contain "rectum". If there is more than one critical (sensitive) structure, the

string "rectum" is replaced by a single quoted string of comma separated organ names, e.g. 'rectum, bladder'. The set "organs" is defined over the critical structure "rectum". The "rungms" interface automatically updates this set if a user excludes organs from consideration as outlined in Section 4.

All necessary data for the optimization is stored (by "gendata") in GAMS GDX format. Therefore, the next component is written to retrieve the data in the GAMS file:

```
$GDXIN data.gdx
$LOAD  PTV=%target% prostate rectum
$LOAD  Dose
$GDXIN
```

"$GDXIN data.gdx" opens the GDX file "data.gdx" for reading. The second line is used to load sets from "data.gdx". Note that a set can be renamed at this stage: the set "target" is now named "PTV". The last line "$GDXIN" closes the file "data.gdx".

Finally, a set "Sensitive" of the sensitive structures is defined as a collection of "allorgans" in the GAMS file. Each organ must be defined explicitly as shown in the second line below:

```
set Sensitive(I,J,k,allorgans);
Sensitive(I,J,K,'rectum') = yes$rectum(I,J,K);
```

Note that gendata forms this file for a given input of the types shown in Appendix A.

C qp.gms

```
* qp.gms
* This program solves 3D conformal radiation treatment problem.
* The solution includes: optimal beam weights
option    limrow=0, limcol=0, solprint=off;
scalar    theta   'dose level prescribed for target'          /1.0/;
scalar    thetaU  'hot spot control parameter on the target'  /1.07/;
scalar    thetaL  'cold spot control parameter on the target' /0.9/;
scalar    ubar    'dose upper bound on the target voxels'     /1.15/;

*---  Read in dose matrix and set definition of the model ---*
$include  initdata.gms
```

```
*---  Define parameters  ---*
parameter phi(allorgans) 'hot spot control parameters for organs';
phi(allorgans) = 0.3;
parameter wup(nAngle)       'upper bound of w(nAngle)';
wup(nAngle) = inf;
parameter DoseUp(allorgans) 'dose upper bound on organ structures';
DoseUp(allorgans) = inf;
parameter DoseLo(allorgans) 'dose lower bound on organ structures';
DoseLo(allorgans) = -inf;
parameter lambda(allorgans) 'objective function penalty parameters';
lambda(allorgans) = 1;

*---  OPTIMIZATION MODEL DEFINITION  ---*
positive variables      w(nAngle), dS(I,J,K,allorgans);
variable                sumDose(I,J,K), z;

equations
  Def4sens(I,J,K,allorgans),
  Def4sumDose(I,J,K),
  Obj;


Def4sumDose(I,J,K)$(PTV(I,J,K) or OAR(I,J,K) or Normal(I,J,K)) ..
  sumDose(I,J,K) =e= sum(angle,Dose(I,J,K,angle)*w(angle));

Def4sens(OAR,organs)$Sensitive(OAR,organs) ..
  -sumDose(OAR) + dS(OAR,organs) =g= -phi(organs);

Obj ..
 z =e=  lambda('%target%')*sum(PTV,sqr(sumDose(PTV)))/card(PTV)
      + sum(organs,lambda(organs)*sum(OAR$Sensitive(OAR,organs),
        sqr(dS(OAR,organs)))/max(1,sum(OAR$Sensitive(OAR,organs),1)))
      + lambda('normal')*sum(Normal,sqr(sumDose(Normal)))
        /max(1,card(Normal));

*--- Bound constraints & Data update ---*
Normal(I,J,K)=yes;
$if exist updatedata.gms  $include updatedata.gms
wup(nAngle)   = min(wup(nAngle),ubar/smax(PTV,Dose(PTV,nAngle)));
w.up(nAngle)  = wup(nAngle);
```

```
angle(nAngle) = yes$(w.up(nAngle) gt w.lo(nAngle));
OAR(I,J,K)    = yes$sum(organs$Sensitive(I,J,K,organs),1);
Normal(PTV) = no;  Normal(OAR) = no; OAR(PTV) = no;

sumDose.up(PTV) = DoseUp('%target%');
sumDose.lo(PTV) = DoseLo('%target%');
sumDose.up(Normal) = DoseUp('normal');
loop(organs,
  sumDose.up(I,J,K)$(Sensitive(I,J,K,organs)) = DoseUp(organs);
);

model conf / all/;
solve conf using nlp minimizing z;

*--- Write out the solution for Matlab users  ---*
sumDose.l(I,J,K) = sum(angle,Dose(I,J,K,angle)*w.l(angle));
$libinclude matout sumDose.l I J K
$libinclude matout w.l nAngle nWedge
```