Solving the dynamic fluence map sequencing problem using piecewise linear leaf position and dose rate functions

Matthew Kelly, Koos van Amerongen, David Craft

Department of Radiation Oncology, Massachusetts General Hospital, Harvard Medical School

November 19, 2017

Abstract

Within the setting of intensity modulated radiation therapy (IMRT) and the fully continuous version of IMRT called volumetric modulated radiation therapy (VMAT), we consider the problem of matching a given fluence map f as well as possible in limited time T by the use of a multi-leaf collimator (MLC). We use linear splines for MLC leaf positions and the dose rate as functions of time, a novel contribution of this work. The optimization computes the dose rate and leaf trajectories that best produce the target fluence map. Computing the dose rate and leaf trajectories is a non-convex optimization problem with many local minima. In our optimization we use a smooth model of how the leaves block the fluence radiation, which helps speed computation and avoid local minima. We solve the optimization in two parts: an outer optimization loop that optimizes the dose rate pattern over time, and an inner optimization loop that, given a fixed dose rate pattern, optimizes the leaf trajectories.

1 Introduction

The optimal dynamic delivery of a given fluence map by a multi-leaf collimator (MLC) remains a difficult, largely unsolved problem. The sliding-window leaf-sweep algorithm (SWLS) [1], in which the MLC leaves cross the treatment field in a unidirectional fashion, achieves perfect fluence map replication if sufficient time is available [2]. However, the SWLS algorithm is not in general efficient with respect to the required delivery time. Time is an important aspect of VMAT and IMRT treatment plans, for several reasons:

i) The effect of patient movement on delivery inaccuracy increases in the time the patient is exposed to radiation.

ii) Shorter treatments allow the treatment facility to help more patients on a given set of radiation therapy machines, which is particularly relevant to third-world countries as these machines are expensive.

iii) In general, there is a trade-off between dose quality and delivery time, and given how widespread the use of radiation therapy is in treating cancer, it makes sense to put in effort to assure that we are on the Pareto optimal frontier regarding these two objectives.

Several studies have investigated the trade-off between treatment time and plan quality [3, 4, 5]. [6] were the first to include treatment time directly in a dynamic leaf sequencing step of the treatment plan optimization. They constructed the trade-off curve between delivery time and fluence map matching accuracy by optimizing leaf trajectories and dose rate patterns for a sequence of delivery times. For a given fluence map and fixed delivery time, the challenge of optimizing the leaf trajectories and dose rate versus time so that the given fluence map is matched as accurately as possible, subject to machine restrictions, presents a high dimensional nonconvex optimization problem. The nonconvexity of the fluence map matching problem leads to a large number of local minima. For a thorough introduction to the complexities of dynamic fluence map delivery (which generally arises in the context of dynamic IMRT and VMAT), see [6] and [7].

2 Model

Our starting point is a fluence map m that has been optimized, along with additional fluence maps located around the patient, to collectively yield a dose distribution optimized for the particular patient's geometry (location of tumor and all nearby organs) and dose prescription. We do not model this aspect of the problem and simply assume that the optimal fluence maps are given. The algorithms set forth in this paper determine how to construct a single given fluence map by moving the leaves of the MLC across the field, while varying the dose rate. Our optimization allows the leaves to move back and forth, a requirement for achieving optimal motions, as shown in the Appendix of [6]. Moreover, we allow the leaves of every pair to start and end separately and not necessarily at the bounds of the treatment field, as these restrictions can also be suboptimal [8].

We assume the fluence map m is given as a matrix where the rows correspond to the individual left and right leaf pairs, and the columns are the discretely optimized fluence bixels across the field, which can be as finely discretized as one wishes. Typical length scales are on the order of 0.5 cm for both the row height of the MLC leaves and the across-the-row discretization.

Let $x_L^i(t)$ and $x_R^i(t)$ denote the leaf position of the ith left and right leaves respectively, at time t. Our framework puts the dose rate optimization in an outer loop. Once the outer loop

sets a dose rate over time profile, the leaf rows can be optimized independently (neglecting the small coupling terms created by the tongue-and-groove mechanism on the real machine, see [7]), so for the remainder of the algorithm development, we consider only a single leaf row.

For the remainder of this section we will consider a single row of the fluence map m, and thus a single pair of leaves (left and right). Let $f(x)$ be the target fluence that should be delivered for that row.

We assume the total allowed treatment delivery time T is given. Our goal is then to compute leaf trajectories $x_L(t)$ and $x_R(t)$ as well as a dose rate $d(t)$ to recreate the fluence row $f(x)$ as best as possible, while accounting for maximum leaf speed, maximum dose rate, and collision constraints.

The fluence achieved at each position is $g(x)$, which is the time-integral of the dose rate for the times that position is exposed to the radiation source. The time domain $\mathcal{T}(x)$ is the set of times (in general a disconnected set) when the position x is not blocked by either of the leaves, i.e., $\mathcal{T}(x)$ is the set of all times t such that $x_L(t) \leq x \leq x_R(t)$, as illustrated by Figure 1 .

$$g(x) = \int_{\mathcal{T}(x)} d(t)dt \tag{1}$$

Our goal is to find the leaf trajectories $x_L(t)$ and $x_R(t)$ and dose rate pattern $d(t)$ that minimize the squared integral error between the target fluence $f(x)$ and the delivered fluence $g(x)$:

$$\underset{d(t),\, x_L(t),\, x_R(t)}{\operatorname{argmin}} \int_X \Big(f(x) - g(x) \Big)^2 dx. \tag{2}$$

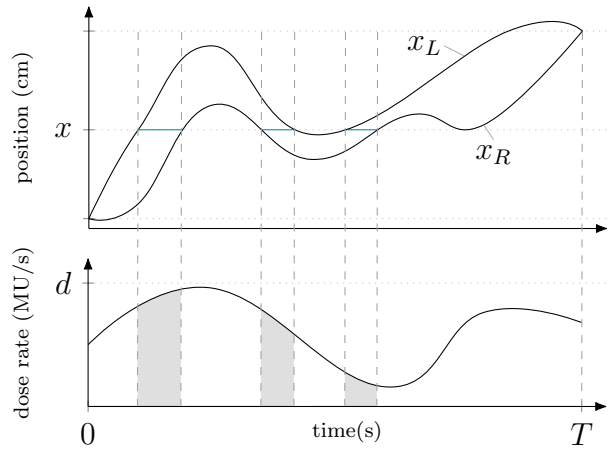

Figure 1: Illustration of administered fluence. In the upper panel, the upper and lower lines display the trajectories of the left and right leaves, respectively; the lower panel shows the dose rate pattern. The dose administered to a position x, $g(x)$, equals the integral (shaded area) of the dose rate $d(t)$ over the moments in time $\mathcal{T}(x)$ (blue lines) that position is exposed.

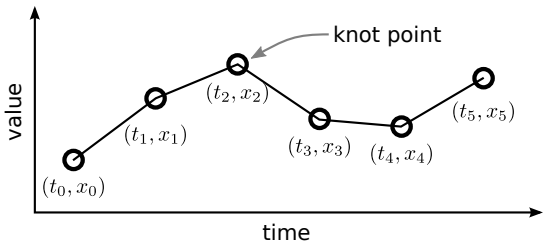

Figure 2: Linear Spline. We represent the dose rate and leaf position trajectories as linear splines. A linear spline is fully defined by its values at the knot points.

3 Method

In this section we describe the method that we use to convert the mathematical optimization problem from Section 2 into a nonlinear program (NLP) that can be solved using standard NLP solvers such as FMINCON [9], SNOPT [10], or IPOPT [11].

The first step in this process is to select a discrete representation for the leaf and dose rate trajectories. Here we will use piecewise linear functions. The second step is to compute the integrals in the objective function using methods that are smooth and consistent, a critical step for obtaining good results from the NLP solver.

3.1 Spline Representation

There are three continuous functions that must be computed by the optimization: the position of the left and right leaves, $x_L(t)$ and $x_R(t)$, and the dose rate $d(t)$. We use piecewise linear functions (linear splines) to represent each function. A linear spline is fully defined by its value at the knot points t_k: $x_{L,k}$, $x_{R,k}$, and d_k. For simplicity we choose to use the same set of knot points for all three splines. An example of a linear spline is shown in Figure 2.

3.2 Integral Computation with Blocking Function $k()$

One critical step in evaluating the objective function (Equation 2) is to compute an approximation of the delivered fluence $g(x)$, given in Equation 1.

There are two numerical issues with computing this integral directly: 1) computing the domain $\mathcal{T}$ requires a root solve (or inverting the leaf trajectories), and 2) the domain of $\mathcal{T}$ can change from being simply connected to discontinuous during an optimization. Both of these issues would likely cause convergence failures in the NLP solver, in part by causing a change in the sparsity pattern of the gradient (*e.g.* $\frac{\partial g}{\partial x_L}$)) between successive iterations.

Our solution is to rewrite the integral using a blocking function $k(t, x)$, which has a value of one when the leaves at time t are passing radiation at location x and zero when the leaves are

blocking radiation. This allows us to rewrite the integral using the constant bounds $[0, T]$:

$$g(x) = \int_0^T k(t, x) \cdot d(t)\, dt \tag{3}$$

We now have a standard scalar integral and we can use any quadrature method to evaluate (3). In our case we use the midpoint (rectangle) quadrature rule.

As just defined, our fluence blocking function $k(t, x)$ would also have a discontinuous gradient, which would cause problems in the optimization. Therefore, we use an exponential sigmoid function to approximate the step function $s(x, \alpha) = (1 + e^{-x\alpha})^{-1}$, where α is the smoothing parameter. A smaller smoothing parameter will provide a more accurate model, while a larger smoothing parameter will lead to faster optimization.

We can then combine the smoothing function for each leaf to get the combined blocking function:

$$k(t, x) \approx \sqrt{s\big(x_R(t) - x,\, \alpha\big) \;\cdot\; s\big(x - x_L(t),\, \alpha\big)} \tag{4}$$

[**TO DO:** *@Matthew, check this equation (see mail)*]

In practice it is useful to define the α in terms of a smoothing distance Δx and the value γ of the smoothing function at a distance Δx from the boundary. For example, $\Delta x = 0.05$ cm and $\gamma = 0.98$ means that at a distance of 0.05 cm from the edge of the leaf, it is blocking 98% of the radiation, and at a distance of -0.05 cm it is blocking 2% of the radiation.

$$\alpha = \frac{-\ln(1 - \gamma)}{\Delta x} \tag{5}$$

[**TO DO:** *@Matthew, check this equation (see mail)*]

3.3 Objective Function

The objective function for the inner optimization (computing leaf trajectories) is the integral of the error-squared between the desired fluence $f(x)$ and the fluence that is delivered by the current set of trajectories, $g(x)$.

$$J = \int_{x_{\min}}^{x_{\max}} \Big(f(x) - g(x)\Big)^2\, dx \tag{6}$$

In practice we can only compute the fluence profile at a finite number of points. We will break the domain $[x_{\min}, x_{\max}]$ into N_{fit} equal-width segments, and evaluate the fluence target and

delivered fluence at the midpoint x_k of each segment.

$$J \approx \frac{x_{\max} - x_{\min}}{N_{\text{fit}}} \sum_{k=1}^{N_{\text{fit}}} \Big(f(x_k) - g(x_k) \Big)^2 \tag{7}$$

3.4 Computing Leaf Trajectories as a Nonlinear Program

The inner optimization loop computes the leaf trajectories $x_L(t)$ and $x_R(t)$ that minimize the objective function (7) and satisfy the position and velocity constraints given below. This optimization is solved as a nonlinear program.

We model the leaf trajectories as piecewise-linear functions of time, where the decision variables in the optimization are the position of each leaf at the knot points in the spline: $x_{L,k}$, $x_{L,k}$, as shown in Figure 2.

We can compute the position on each segment by linear interpolation between the knot points.

The position of each leaf is linear over a single segment, thus the velocity of the leaf on that segment is constant, with a change in value at each knot point. The velocity for each segment is:

$$\dot{x}_{L,k} = \frac{x_{L,k+1} - x_{L,k}}{h_k} \qquad \dot{x}_{R,k} = \frac{x_{R,k+1} - x_{R,k}}{h_k} \tag{8}$$

where h_k is the distance between knot point k and $k+1$ (we use equal spacing for all knot points).

The limits on leaf position can be implemented as a combination of constant bounds and linear inequality constraints:

$$x_{\min} \le x_{L,k} \qquad x_{R,k} \le x_{\max} \qquad x_{L,k} \le x_{R,k} \qquad \forall k \tag{9}$$

The limits on velocity can be written as linear inequality constraints:

$$-v_{\max} \le \dot{x}_{L,k} \le v_{\max} \qquad -v_{\max} \le \dot{x}_{R,k} \le v_{\max} \qquad \forall k \tag{10}$$

At this point we can compute the piecewise-linear position trajectories and the piecewise-constant velocity trajectories, and enforce limits on them inside the non-linear program. The final step is to compute the objective function for the candidate trajectories, which is done as described in Section § 3.2.

3.5 Iterative refinement of smoothing parameter

In practice, the leaf trajectory optimization runs well when the smoothing distance γ is large. As the smoothing becomes smaller, the optimization becomes more difficult to solve, but the model is more accurate.

We can use these properties to our advantage by iteratively solving the optimization problem. On the first iteration we use a large value for the smoothing parameter, which will quickly find a good approximation of the solution. On subsequent iterations we use the solution from the previous optimization as the initial guess, and then reduce the smoothing parameter until we achieve the desired model accuracy.

This process is effective at avoiding local minima because the optimization with heavy smoothing inherently focuses on the global aspects of the problem. Subsequent optimizations are similar, so the previous solution is an excellent guess, and the optimization is able to simply refine that solution for the more accurate model.

3.6 The outer loop: computing the dose rate trajectory

We adopt a direct global search strategy for the outer optimization loop: computing the optimal dose rate trajectory.

We use the CMAES algorithm [12], which works by widely sampling the objective function (dose rate trajectory knot points in our case) and then fitting a multi-variate Gaussian to the value of the objective at those points. Then it samples new points from that Gaussian and updates the model.

Since CMAES does not require explicit gradient calculations, it tends to be more robust to problems with many local minima and difficult objective functions.

4 Results

In this section we present a collection of simple experiments to demonstrate various features of the proposed algorithm.

[**TO DO:** *Recompute the plots in this section to be consistent with new definition of the smoothing parameter.*]

[**TO DO:** *@Matthew, think of a figure to compare different smoothing configurations*]

4.1 Smoothing Comparison

In Section §3.2 we describe a smooth model of how the leaves block radiation. Here we describe a simple experiment to look at how adjusting that smoothing parameter affects the optimization.

We used a single target fluence profile, as shown in Figure 4, and assumes that the dose rate is a constant function of time. We performed a simple grid search, computing the optimal

leaf trajectories for each dose rate value. We then ran this grid-search twice, once with heavy smoothing and once with light smoothing.

The smoothing parameter can be expressed in terms of a characteristic blurring width. Here we used a width of 0.5 cm for the heavy smoothing and 0.05 cm for the light smoothing. The characteristic width was computed such that the smoothing function achieved 95% of its change in value over the smoothing window.

We used seven of leaf trajectory segments, although similar results are obtained for trajectories with more segments. This number was computed with a pilot study. Fewer trajectory segments lead to faster solve times, but the fitting error increases significantly. Using more trajectory segments takes longer, but at a trivial reduction in fitting error.

[**TO DO:** *add a figure for number of segments? Or maybe another sub-section?*]

Figure 3 shows the two sequences of optimizations, one for the heavy smoothing of 0.5 cm and the other for the light smoothing of 0.05 cm. The objective function is normalized by the fitting error associated with a solution where the leaves completely block the radiation. Although both optimizations are able to find reasonable solutions, there are a few salient differences. The optimization with light smoothing takes about four times longer to run when compared to the optimization with heavy smoothing. The optimization with light smoothing also tends to get stuck in local minima, as shown by the objective function jumping around with small changes in the constant dose rate. The optimization with heavy smoothing eventually converges to trajectories that rely on the smoothing behavior to achieve the desired fluence profile. This is clear from the discrepancy between the smooth and exact model for dose rates above 4 MU. That being said, this discrepancy between the models is small and the overall shape of the delivered fluence map is generally correct, as shown in Figure 4.

4.2 Iterative Smoothing Refinement

[**TO DO:** *MPK: Do we plan to keep all four figures in this subsection? Either way, I can spend a bit of time to make the text more concise. My guess is that we could do with a single of the 'example' figures.*]

One common technique in trajectory optimization is to solve a trajectory optimization problem iteratively, making small adjustments to the problem statement between each optimization, and using the result of the previous optimization to seed the next.

We can do this iterative refinement with the leaf-blocking smoothing parameter, starting with heavy smoothing and then using that solution to seed another optimization with a smaller smoothing term.

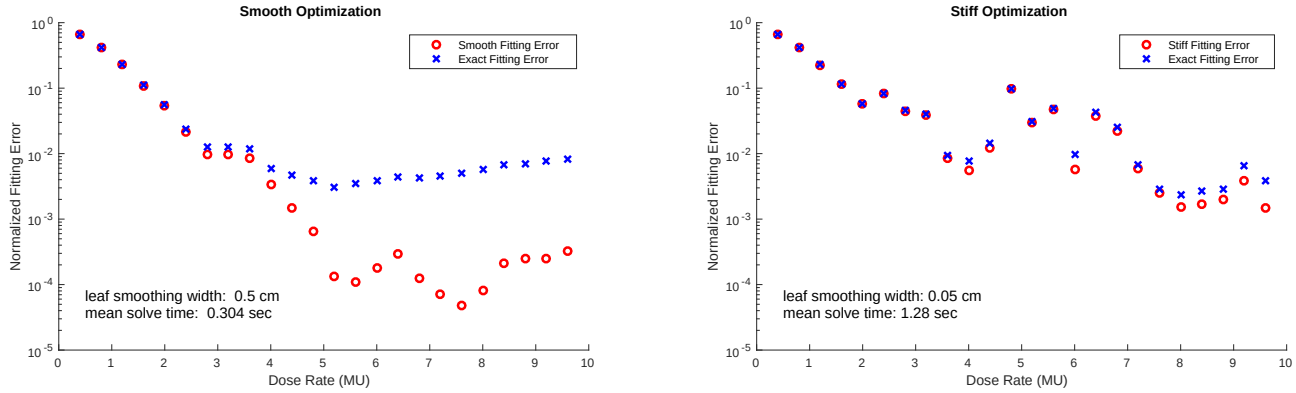

Figure 3: Comparison of heavy and light smoothing in leaf-blocking model. In each plot we ran 24 leaf optimizations, sweeping through a range of constant dose rate trajectories. The optimizations in the left plot used a leaf smoothing distance of 0.5 cm, while the right plot used 0.05 cm. Notice that the left set of optimization does a better job of avoiding local minima, but that there is a disparity between the fluence profile produced by the smooth model and the exact (no smoothing) model. The right plot does better job of matching the exact model, but it tends to get stuck in local minima and takes significantly (4x) longer to run. The solution for a dose rate of 5.2 MU on the left plot is shown in detail in Figure 4.

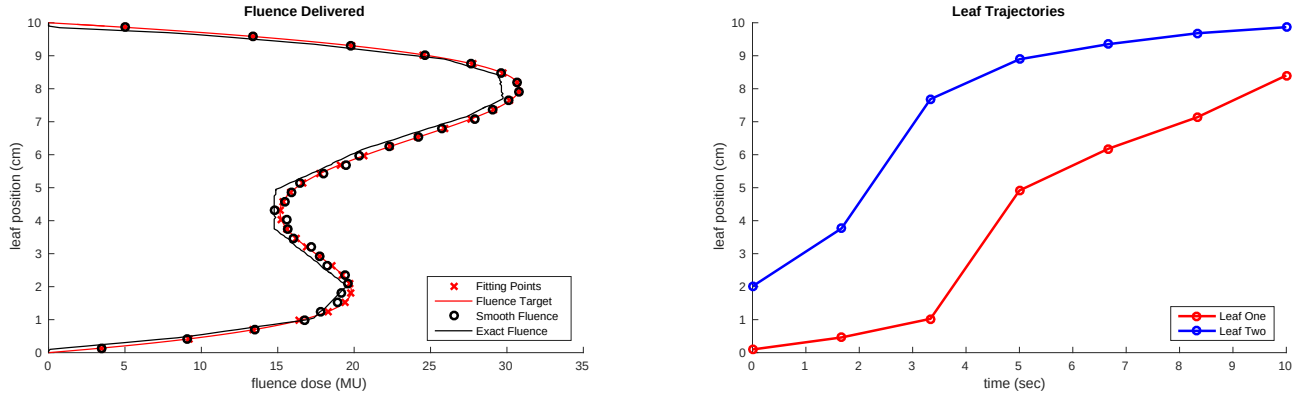

Figure 4: Optimal leaf-trajectories, computed with a smoothing width of 0.5 cm and a constant dose rate of 5.2 MU. Note the small discrepancy between the smooth model of the delivered fluence and the exact (no smoothing) model.

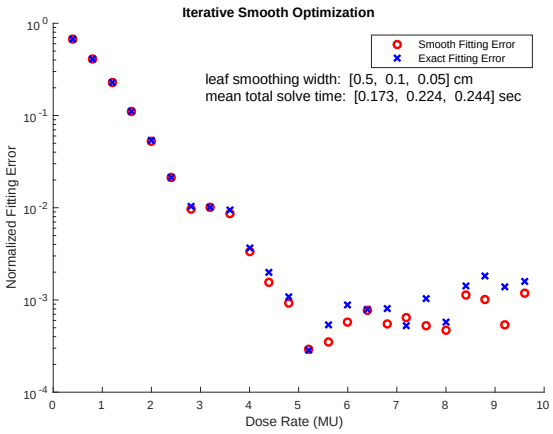

Figure 5: This plot shows the objective function value for the optimal leaf trajectories, given a constant fluence dose rate. Each optimization here is actually a set of three optimizations, each using a successively smaller smoothing distance. In all cases the first optimization used a smoothing distance of 0.5cm, the second used 0.1cm, and the final optimization used 0.05cm. This technique results in faster solve times than a single optimization with the smallest smoothing parameter, while also avoiding local minima.

We test this procedure by doing an experiment similar to the one discussed in §Section 4.1, but using the solution of the heavy-smoothing optimization to initialize the optimization with light smoothing. Here we will use a sequence of three optimizations, starting with a smoothing of 0.5 cm, then moving to 0.1 cm, and finally 0.05 cm for the final optimization.

The resulting sweep of optimizations is shown in Figure 5. These results are better than either the heavy or light smoothing optimizations. The total optimization time for all three optimizations is less than simply running the light smoothing optimization from a niave guess, the model discrepancy is tiny, and the solver is generally able to do a good job of avoiding local minima.

Figure 6 again shows the optimal leaf trajectories with a constant dose rate of 5.2 MU. The leaf trajectories are qualitatively similar to those in Figure 4, but slight changes are present that make the exact-model fluence profile much closer to the target profile.

4.3 Computing Dose-rate Trajectories

[**TO DO:** *CMAES optimization for a single trajectory?*]

4.4 Multi-VMAT

[**TO DO:** *David - should we try to run an optimization with multiple pairs of leaves?*]

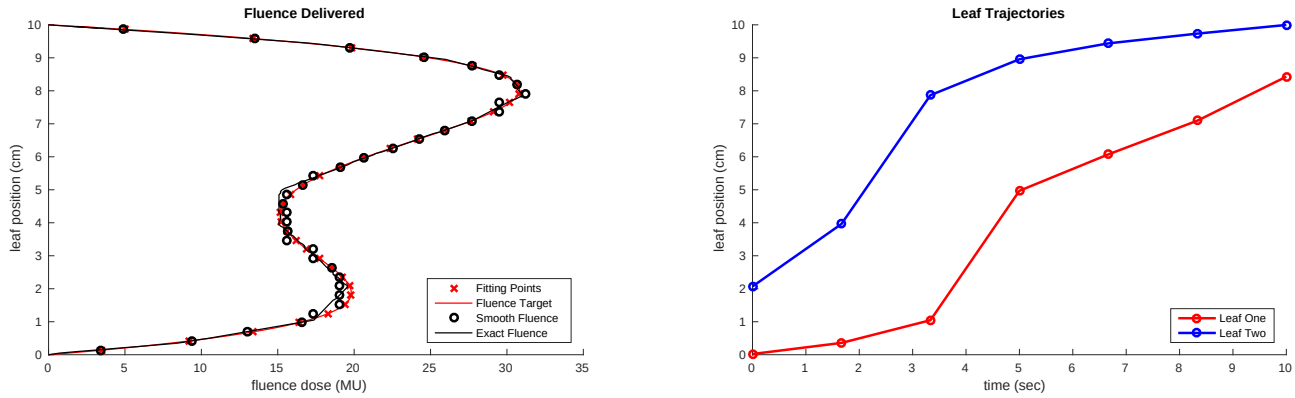

Figure 6: This plot shows the optimal leaf trajectories and delivered fluence map using the best solution (dose rate = 5.2 MU) found by the iterative optimization procedure. Note that these trajectories are similar to those in Figure 4, but that the fitting is better here.

4.5 Misc. Notes:

[**TO DO:** *Clean up this section, perhaps add a few small plots.*]

It seems that the velocity smoothing has little effect on the optimization: no significant change observed in computation time, convergence, or trajectories.

The number of fitting points is important – if there are too few the trajectories tend to become less smooth and more local minima pop up. I believe that this is caused by overfitting.

The solve time varies linearly with number of trajectory segments, at least on the range of 7 to 13 segments. The solution quality does not appear to change that much, although a few extra bumps do appear in the trajectories. If there are too few points, then the it is sometimes not possible to capture some features in the target profile.

[**TO DO:** *Decide which of these points we include and which we drop.*]

5 Discussion and conclusions

5.1 Why first-order splines to represent trajectories?

In this section we informally discuss some of the design choices that were made when creating this trajectory-based method for solving leaf and dose rate trajectories for fluence mapping.

[**TO DO:** *This section should be edited to reduce length and needs some detailed citations.*]

There are many ways to represent trajectories, and nearly all of them have associated transcription (optimization) methods. In general there is one major trade-off: use a larger number of low-order segments or use a smaller number of high-order segments.

Selecting the correct method order typically comes down to problem specifics. High-order methods are best when the system model is good and high accuracy is important. The accuracy of these methods relies on the underlying problem being smooth over each mesh interval, with few places where path constraints serve to shape the trajectory. Low-order methods are best when the trajectory shape is dominated by path constraints and high accuracy is not critical.

[**TO DO:** *Cite my tutorial paper? Check if I discuss this in detail. Also see if it is discussed in the reviews by Rao or Betts.*]

Although our specific problem does not have path constraints, it does have a highly nonlinear objective function. The smooth leaf-blocking model causes a strong nonlinearity in the objective function, which would make careful mesh refinement necessary if high-order methods were used.

[**TO DO:** *Cite GPOPS-II paper? Or perhaps Rao or Betts tutorial paper.*]

We can avoid the need for careful mesh refinement by simply using a low-order model, which also makes it easier to precisely handle the velocity and position limits on the leaf trajectories.

We did a few brief checks, comparing a few different schemes using cubic splines and compared them to the linear spline presented in this paper. Although both methods worked, the linear spline methods were much faster and were able to provide an good fit to the desired fluence profile. One possible reason for the speed increase is that a set of three linear segments have less coupling terms than a cubic spline fitting the same trajectory domain. The resulting sparsity in the jacobian (derivative of the objective function) helps optimization. [13]

The major drawback of a linear spline for leaf trajectories is that the resulting trajectory tends to be less smooth, with a step change in velocity between each segment. This effect can be minimized by using a large number of linear segments. In some cases this can lead to a slight numerical instability, which is easily addressed by adding a small regularization term to the optimization, minimizing the integral of the velocity-squared along the trajectory.

5.2 Inner Optimization: Leaf Trajectories

[**TO DO:** *Discuss figures in results section.*]

5.3 Outer Optimization: Dose-Rate Trajectories

[**TO DO:** *Try doing a better job of performing two optimizations, and varying the leaf smoothing between them. See if this improves results.*]

The outer optimization with CMAES works, but it still has the problem with local minima: running the optimization N times produces several different solutions. These solutions all have similar objective function values, but are created by different dose rate trajectories.

It seems to me that the dose rate optimization has a large number of similar-valued minima, since the leaf-trajectory optimization can find good solutions for most dose rate trajectories. The residual fitting error seems to be more related to the choice of discretization – number of grid segments, rather than convergence failure.

I see two ways to go forward from here. First would be to try this optimization with a set of leaf trajectories, to see if that helps force a better global solution, since the problem will be more constrained. Second would be to play around more with global optimization and smoothing terms.

If I make the smoothing term on the dose rate trajectory large, then I get repeatable solutions, but they tend to be nearly-constant dose rate, and often relatively high.

Another option would be to modify the regularization term to include a penalty on large dose rates, which is perhaps desirable in the real system.

References

[1] D. Convery and M. Rosenbloom. The generation of intensity-modulated fields for conformal radiotherapy by dynamic collimation. *Physics in Medicine and Biology*, 37(6):1359, 1992.

[2] Jörg Stein, Thomas Bortfeld, Birgit Dörschel, and Wolfgang Schlegel. Dynamic x-ray compensation for conformal radiotherapy by means of multi-leaf collimation. *Radiotherapy and Oncology*.

[3] E. Salari, J. Wala, and D. Craft. Exploring trade-offs between VMAT dose quality and delivery efficiency using a network optimization approach. *Physics in Medicine and Biology*, 57(17):5587, 2012.

[4] D. Craft, T. Hong, H. Shih, and T. Bortfeld. Improved planning time and plan quality through multicriteria optimization for intensity-modulated radiotherapy. *Int. J. Radiation Oncology Biol. Phys.*, 82(1):83–90, 2012.

[5] D. Craft and T. Bortfeld. On the tradeoff between treatment time and plan quality in rotational arc radiation delivery. *arXiv preprint arXiv:0910.4934*.

[6] M. Balvert and D. Craft. Fast approximate delivery of fluence maps for IMRT and VMAT. *Physics in Medicine and Biology*, 62(4):1225, 2017.

[7] J. Unkelbach, T. Bortfeld, D. Craft, M. Alber, M. Bangert, R. Bokrantz, D. Chen, R. Li, L. Xing, C. Men, S. Nill, D. Papp, E. Romeijn, and E. Salari. Optimization approaches to volumetric modulated arc therapy planning. *Medical Physics*, 42(3):1367–1377, 2015.

[8] J.H.M. van Amerongen. Fast approximate delivery of fluence maps in volumetric modulated arc therapy. Master's thesis, Tilburg University, The Netherlands, 2017. Unpublished.

[9] Mathworks. Matlab Optimization Toolbox, 2014.

[10] Large-scale Nonlinear Programming, Philip E Gill, Walter Murray, and Michael A Saunders. User ' s Guide for SNOPT Version 7 : Software for. pages 1–116, 2006.

[11] Andreas Wächter and Lorenz T. Biegler. *On the implementation of primal-dual interior point filter line search algorithm for large-scale nonlinear programming*, volume 106. 2006.

[12] N Hansen and a Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary computation*, 9(2):159–95, jan 2001.

[13] John T. Betts. *Practical Methods for Optimal Control and Estimation Using Nonlinear Programming*. Siam, Philadelphia, PA, 2010.