Dynamic leaf sequencing via trajectory representations of leaf trajectories and dose rate

Matthew Kelly, Koos van Amerongen, David Craft

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

August 7, 2017

Abstract

We consider the problem of matching a given fluence map f to a close approximation in limited time T by the use of a multi-leaf collimator, i.e. the setting of IMRT and VMAT in radiation therapy. We use linear splines for leaf position and the dose rate as functions of time. The optimization computes the dose-rate and leaf trajectories that produce the best match for the target fluence map. Computing the dose-rate and leaf trajectories is a difficult optimization problem with many local minima. In our optimization we use a smooth model of how the leaves block the fluence radiation, which help speed computation and avoid local minima. We solve the optimization in two parts: an inner optimization computes the optimal leaf trajectories, given a dose-rate trajectory, and an outer optimization computes the dose-rate trajectory that minimizes the fitting error across all inner leaf-trajectory optimizations.

1 Introduction

The optimal dynamic delivery of a given fluence map remains a difficult, largely unsolved problem, due to its inherent nonconvexity. The nonconvexity of the fluence map matching problem leads to a large number of local minima. Many methods simply ignore this crucial aspect of the problem, and others use a multi-start procedure to find several local minima and return the best one found. For a thorough introduction to the complexities of dynamic fluence map delivery (which generally arises in the context of volumetric modulated arc therapy, VMAT), see [1] and [2].

In this technical note, we investigate a regularization procedure which represents leaf trajectories and dose rates as linear splines (piece-wise linear functions), such that the optimization procedure naturally focuses on global aspects of the leaf trajectory/dose rate solution.

1.1 Background

[**TO DO:** *David – Would you be able to add any background that is appropriate for this paper, at least from a fluence mapping perspective? Let me know if I should add a short section about trajectory optimization. Also, should background be its own section, or is a sub-section in the Introduction acceptable?*]

1.2 Fluence Mapping as Continuous Optimization

Our goal is to compute the optimal leaf trajectories $z_L(t)$ and $z_U(t)$, as well as the optimal dose rate $r(t)$ of the radiation source. For this initial study we will consider only a single leaf pair, although our algorithm is structured such that it could be applied to an arbitrary number of leaf pairs. We will assume that adjacent leaves are independent of each other, neglecting the small coupling terms created by the tongue-and-groove mechanism on the real machine.

We will assume that the desired fluence dose at each point $q_D(x)$ on the subject is given, and that we would like to compute the dose-rate $r(t)$ and leaf trajectories $z_L(t)$ and $z_U(t)$ that minimize the error between the fluence that is predicted to reach the target $q(x)$ and the desired fluence $q_D(x)$.

$$\underset{r(t),\, x_L(t),\, x_U(t)}{\operatorname{argmin}} \int_X \Big(q(x)) - q_D(x) \Big)^2 dx \tag{1}$$

The fluence dose at each location is the time-integral of the dose rate trajectory for the times when the radiation is not being blocked by the upper or lower leaves. This time domain $T(x)$ is the set of times when the position x is not blocked by the leaves: $T(x)$ is the set of all times t such that $x_L(t) < x < x_U(t)$.

$$q(x) = \int_{T(x)} r(t) dt \tag{2}$$

For this work we will leave the initial and final leaf positions as decision variables in the optimization, but it is a trivial extension to enforce arbitrary initial and final leaf positions with a simple modification to the position limits at these points in the trajectory. This might be useful when combining a sequence of fluence maps as shown in **[REF Multi-VMAT paper]**.

We will also assume that the total duration of the trajectories is given, but this could easily be added as a decision variable at a later time.

1.3 Implementation as a Nested Optimization

The optimization formulation presented in Section §1.2 is a good mathematical description of our goal, but it is not obvious how to solve it. In this paper we use a nested optimization approach, solving the dose-rate trajectory $r(t)$ in an outer optimization, and then solving the optimal leaf trajectories $z_L(t)$ and $z_U(t)$ as an inner optimization, given a candidate dose-rate trajectory.

Partitioning the problem in this fashion makes physical sense: given a fixed dose-rate trajectory, the leaf trajectories are completely independent. This decoupling allows us to focus on designing the inner and outer optimizations differently, each specialized for the details of that optimization.

[**TO DO:** *David - do we need an equation here? If so, I'm not sure of the best way to write the nested optimization.*]

2 Method

In this technical note we focus on developing an algorithm for reliably computing the optimal leaf trajectories, given a candidate dose-rate trajectory. We use linear splines for the dose-rate and leaf trajectories, and avoid many of the issues with local minima by using a smooth model of how the leaves block radiation.

The inner optimization (leaf trajectories) is transcribed to a smooth non-linear program (NLP), which can easily be solved using standard NLP solvers such as FMINCON [3], SNOPT [4], or IPOPT [5].

The outer optimization is still a difficult optimization, which is made somewhat non-smooth by having an iterative optimization procedure inside the objective function. As a result, it is likely that smooth solvers such as those used for the inner optimization would have difficulty converging due to bad gradient estimates. For the purposes of this technical note we decided to use CMAES [6], a good general-purpose solver for difficult non-linear optimization problems.

2.1 Trajectory Representation

The final goal of our optimization is to compute three trajectories: radiation dose-rate $r(t)$, lower leaf position $z_L(t)$, and upper leaf position $z_U(t)$.

As discussed in Section §3.1, each of these trajectories is represented by a piecewise linear spline. Each of the three trajectories shares the same set of knot times $t_k \in \{t_0, t_1, \ldots, t_N\}$, where N is the number of knot points. The value of the dose rate and leaf positions at knot time t_k are given by r_k, $x_{L,k}$, and $y_{U,k}$ respectively. We assume that the total duration of the trajectory

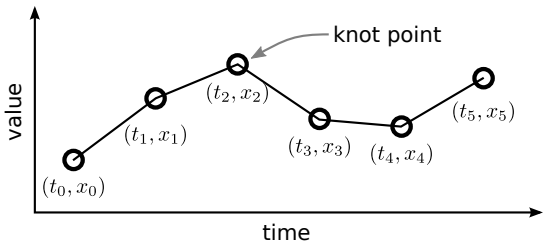

Figure 1: Linear Spline. We represent the dose-rate and leaf position trajectories as linear splines. A linear spline is fully defined by its value at the knot points.

is given, although this could easily be made a decision variable in the outer optimization when computing dose rate.

2.2 Trajectory Limits

There are several constraints imposed on the dose-rate and leaf trajectories by the physical limitations of the VMAT machine. Specifically, there is a maximum dose-rate, constant limits on the leaf positions, and an upper limit on the leaf speed. These limits are detailed below, where $\dot{z}(t) = \frac{d}{dt}z(t)$ gives the leaf velocity.

$$\text{dose-rate:} \quad 0 \leq r(t) \leq r_{\max} \tag{3}$$

$$\text{leaf position:} \quad x_{\min} \leq z_L(t) \leq z_U(t) \leq x_{\max} \tag{4}$$

$$\text{lower leaf velocity:} \quad v_{\min} \leq \dot{z}_L(t) \leq v_{\max} \tag{5}$$

$$\text{upper leaf velocity:} \quad v_{\min} \leq \dot{z}_U(t) \leq v_{\max} \tag{6}$$

2.3 Avoiding Local Minima

One of the key issues with fluence mapping is that the optimization tends to get stuck in local minima, since there are a large number of leaf trajectories that deliver the same fluence to the target. **[REF local minima in fluence mapping]**

It is possible to help the optimization avoid local minima by smoothing the optimization problem. This smoothing is useful because it means that the gradient estimates in the optimization change less between each iteration in the optimization.

There is an obvious trade-off here: smoothing makes the optimization easier by fundamentally changing the problem statement, thus we get an easy optimization that is inaccurate. The solution here is to iteratively reduce the smoothing terms, initially solving the optimization with heavy smoothing to avoid local minima, and then using that solution to initialize another optimization with less smoothing. See §Section 2.5 for details.

This technique of iteratively reducing a smoothing term is widely used in trajectory optimization, such is in [7].

2.4 Computing Delivered Fluence

The fluence delivered at each position on the target is computed by integrating the dose-rate function over the periods of time when the leaves are not blocking the target:

$$f(x) = \int_{\mathcal{T}(x)} r(t)\,dt \qquad \mathcal{T}(x) = \forall t \quad \text{S.T.} \quad z_L(t) \leq x \leq z_U(t) \tag{7}$$

There are two issues with computing this integral directory. The first is that it requires either computing the inverse of the leaf trajectories, which ultimately reduces to a non-linear root finding sub-problem. When placed inside of an optimization, this will dramatically slow convergence. The second issue is that the set $\mathcal{T}(x)$ does not smoothly vary with respect to the leaf trajectories. It is possible to construct a set of leaf trajectories for which a small change in leaf position switches $\mathcal{T}(x)$ from one to two simply connected sets. Inside of an optimization, this would cause a change in the sparsity pattern of the gradient (*e.g.* $\frac{\partial f}{\partial z_L}$)) between successive iterations, which leads to poor convergence and possibly a numerical instability.

Our solution is to rewrite the integral using a blocking function $k(t)$, which has a value of one when the leaves are passing radiation and zero when the leaves are blocking radiation, as described in §2.5 This allows us to rewrite (7) using the constant bounds $[0, T]$:

$$f(x) = \int_0^T k(t,x) \cdot r(t)\,dt \tag{8}$$

Now we have a standard scalar integral, where the integrand is smooth and the bounds are constant, we can use any quadrature method to evaluate (8). In this case we use the midpoint (rectangle) quadrature rule.

2.5 Modeling Fluence Blocking

A simple way to define the fluence blocking function $k(t)$ would be to set it to one if $z_L(t) \leq x \leq z_U(t)$ is true, and zero otherwise. This implementation would have a discontinuous gradient, which would cause problems in the optimization. Instead, we use exponential smoothing to approximate the step function $s(x, \alpha)$, where α is a small smoothing parameter. A smaller smoothing parameter will provide a more accurate model, while a larger smoothing parameter will lead to faster optimization.

$$k(t,x) = \sqrt{s\big(x - z_L(t),\, \alpha\big) \;\cdot\; s\big(z_U(t) - z,\, \alpha\big)} \tag{9}$$

$$s(t, \alpha) = \frac{1}{1 + e^{-t\alpha}} \tag{10}$$

In practice it is useful to define the smoothing parameter α in terms of a smoothing distance Δx and the 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.

$$\alpha = \frac{-\ln(1 - \gamma)}{\Delta x} \tag{11}$$

2.6 Objective Function

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

$$J = \int_{x_{\min}}^{x_{\max}} \Big(f_D(x) - f(x) \Big)^2 \, dx \tag{12}$$

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. This is equivalent to approximating (12) using rectangle (mid-point) quadrature with N_{fit} segments.

$$J \approx \frac{x_{\max} - x_{\min}}{N_{\text{fit}}} \sum_{k=1}^{N_{\text{fit}}} \Big(f_D(x_k) - f(x_k) \Big)^2 \tag{13}$$

2.7 Computing Leaf Trajectories as a Non-linear Program

The inner optimization loop computes the leaf trajectories $z_L(t)$ and $z_U(t)$ that minimize the objective function (12) and satisfy the constraints (3–6).

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

$$x_{\min} \le x_{L,k} \qquad z_{U,k} \le z_{\max} \qquad z_{L,k} \le z_{U,k} \qquad \forall k \tag{14}$$

The trajectories as piecewise linear splines, represented here by their values at the knot points t_i. The velocity trajectory can easily be computed by taking the derivative of the spline as shown below.

$$\dot{z}_{L,k} = \frac{z_{L,k+1} - z_{L,k}}{h_k} \qquad \dot{z}_{U,k} = \frac{z_{U,k+1} - z_{U,k}}{h_k} \tag{15}$$

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

$$v_{\min} \leq \dot{z}_{L,k} \leq v_{\max} \qquad v_{\min} \leq \dot{z}_{U,k} \leq v_{\max} \qquad \forall k \tag{16}$$

We can now construct a non-linear program (NLP) to compute the optimal choice of leaf trajectories, given the dose rate trajectory. The decision variables in the NLP are the values of the two leaf splines at each knot point: $z_{L,k}$ and $z_{L,k}$ for $k \in 0 \ldots N$. The objective function is given by ()13) and the constraint functions by (14) and (16).

2.8 Iterative Refinement of Smoothing Parameter

In practice, the leaf trajectories are easy to compute (§2.7) when the smoothing distance γ (11) 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 reasonably effective at avoiding local minima: the optimization with heavy smoothing has few issues with local minima, finding a good global solution. 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.

2.9 Computing the Does-Rate Trajectory

Using the methods in Section §2.7 we can reliably compute leaf trajectories, given a dose rate trajectory.

There are many ways to compute the dose rate trajectory, but here we use CMAES [6]. CMAES works by widely sampling the objective function (dose-rate trajectories 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 complex objective functions.

[**TO DO:** *Fill out this section in more detail and edit better.*]

3 Discussion

3.1 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. [**?**]

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.

3.2 Inner Optimization: Leaf Trajectories

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

3.3 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.

[**TO DO:** *final results pending*]

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.*]

4.1 Smoothing Comparison

In Section §2.5 we describe a smooth model of how the leaves block radiation. Here we describe do a simple experiment to look at how adjusting that smoothing parameter affects the optimization.

We uses a single target fluence profile, as shown in Figure 3, 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 also used seven of leaf trajectory segments, although similar results are obtained for trajectories with more segments.

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

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.

Figure 2 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 2.

4.2 Iterative Smoothing Refinement

One common trick 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

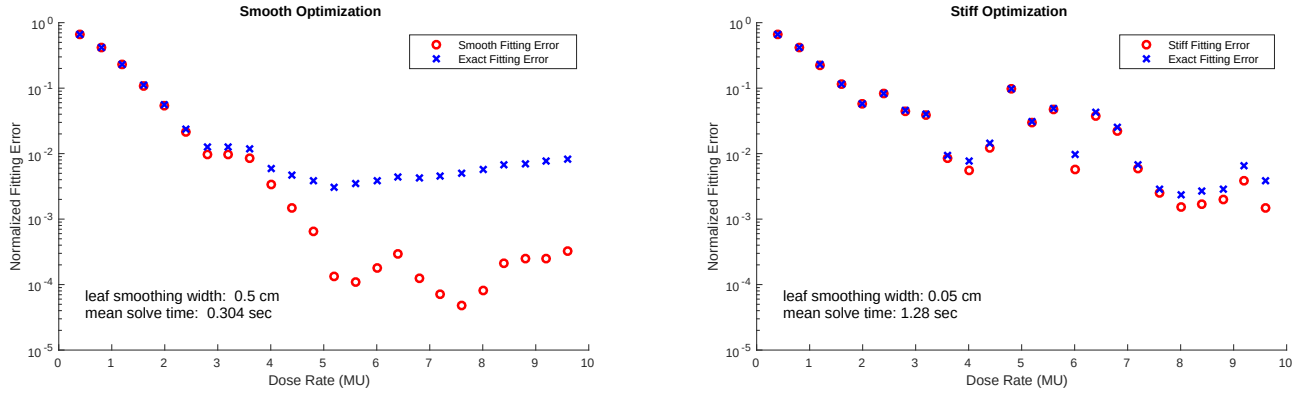

Figure 2: Comparison of heavy and light smoothing in leaf-blocking model. In each plot we ran 24 leaf optimizations, sweeping through a range of constant dose-rate trajectories. The optimizations in the left plot used a leaf smoothing distance of 0.5 cm, while the right plot used 0.05 cm. Notice that 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 3.

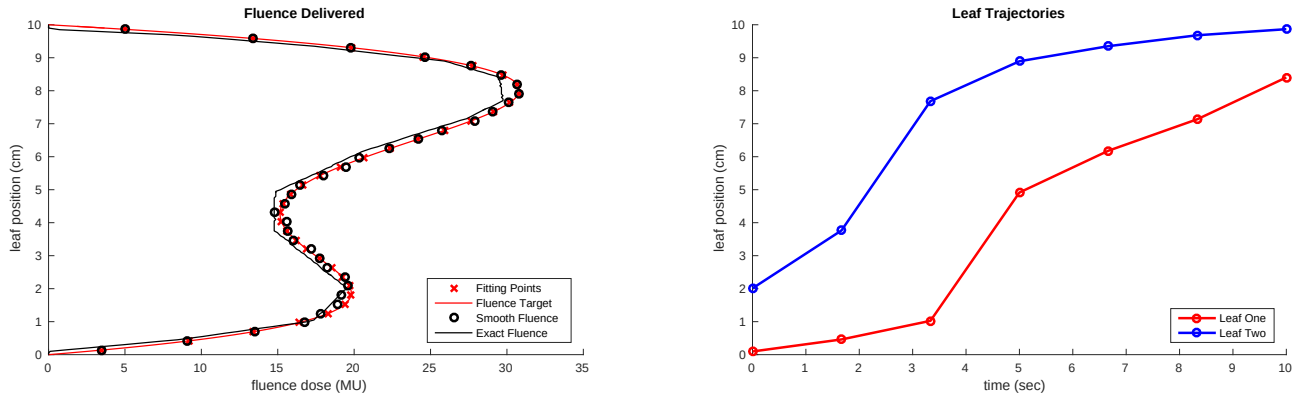

Figure 3: Optimal leaf-trajectories, computed with a smoothing width of 0.5 cm and a constant dose-rate of 5.2 MU. Note the small discrepancy between the smooth model of the delivered fluence and the exact (no smoothing) model.

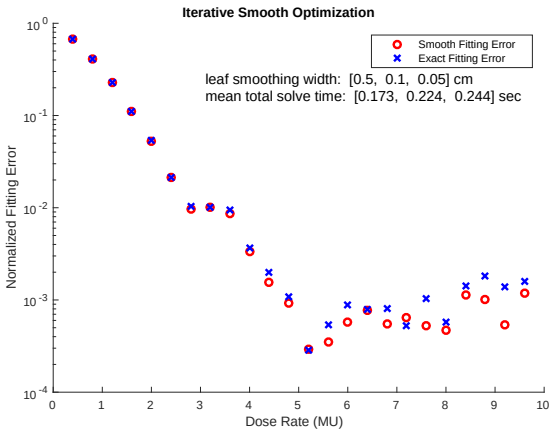

Figure 4: This plot shows the objective function value for the optimal leaf trajectories, given a constant fluence dose rate. Each optimization here is actually a set of three optimizations, each using a successively smaller smoothing distance. In all cases the first optimization used a smoothing distance 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.

with heavy smoothing and then using that solution to seed another optimization with a smaller smoothing term.

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 4. 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 5 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 3, 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?*]

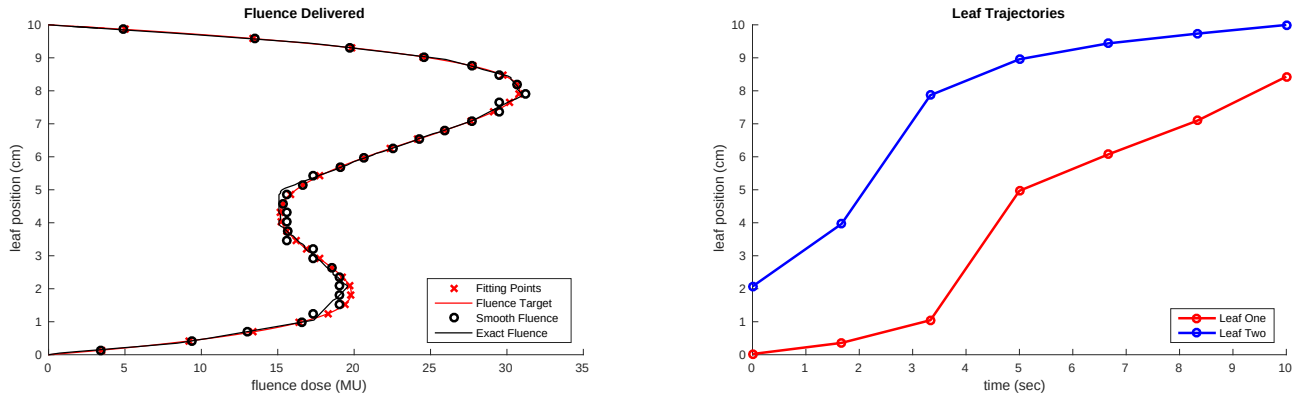

Figure 5: This plot shows the optimal leaf trajectories and delivered fluence map using the best solution (dose rate = 5.2 MU) found by the iterative optimization procedure. Note that these trajectories are similar to those in Figure 3, but that the fitting is better here.

4.4 Multi-VMAT

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

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 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.

5 Results

5.1 Future Work

Automatically determine the number of grid segments.

Multi-leaf trajectory optimization.

Continuous rotation VMAT stuff?

References

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

[2] 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.

[3] Mathworks. Matlab Optimization Toolbox, 2014.

[4] 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.

[5] 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.

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

[7] Manoj Srinivasan and Andy Ruina. Computer optimization of a minimal biped model discovers walking and running. *Nature*, 439(7072):72–5, jan 2006.