Last updated: December 18, 2020

I evaluated convergence of the optimization routine using the following approach. The value function for optimization is

$$\text{value} = \text{objective} - \text{weight}_\text{p} \times \text{penalty} - \lambda \sum(\text{wiggly}^2) \tag{1}$$

This is the objective in the optimal control problem, minus the penalty calculated during optimization (times a penalty weight), minus a penalty for the tortuosity of the solution (how wiggly the curve is). We chose to start with a large weight for the optimization penalty ($\text{weight}_\text{p} = 10$) and large weight for the tortuosity penalty ($\lambda = 1$). We then ran the following routine:

- Use the Runge-Kutta-4 method (rk4) for ODEs and Nelder-Mead for optimization (max 5000 iterations).
- Reset the controls to lie within their constraints.
- Use the implicit Adams method (impAdams_d, or impAdams) for ODEs and BFGS for optimization (max 1000 iterations).
- Reset the controls to lie within their constraints.
- Enter an optimization loop. Within the loop:

  Use the implicit Adams method (impAdams or impAdamsd) for ODEs and Nelder-Mead for optimization (max 2500 iterations).

  Reset the controls to lie within their constraints.

  Use the implicit Adams method (impAdams or impAdamsd) for ODEs and BFGS for optimization (max 1000 iterations).

Reduce the weight for lambda by half.

GS: Is there a reason we might want to reduce the weight for constraint violation by half as well?

I initially tested for how to optimize on the problem for unbranched, determinate plants with a objective function of $\log(F)$, a Uniform(0,5) season length distribution, initial states of [1,.1,0,.0001] and meristem constraints of [.75,1]. I experimented with modifying the following variables: number of iterations of the optimization routine, number of Nelder-Mead iterations, progression of penalty on tortuosity (λ values).

Try removing the constraints from the for loop (SPE). I did this because the value of the optimization routine decreased over time but had some bumps where it decreased then increased again. Steve suggested this might be because of the constraints in the for-loop. I did that but it seemed to really change the solution. So I tried to break the problem apart a bit more.

At each iteration, I kept the (1) value function from optimization, (2) the objective function, (3) the penalty for violating constraints, and (4) the difference of (-1 - (2 - 3)) which I think should approximate the penalty for tortuosity. Dips in the value function were associated with jumps in the objective function but also with the penalty for violating constraints. So I'm going to try running this again, for longer, without any constraints and see what that looks like.

Graph the quadratic vs. 1.25 loss function:

```
> plot(seq(-1,1,by=0.01),seq(-1,1,by=0.01)^2,type='l')
> lines(seq(-1,1,by=0.01),10*seq(-1,1,by=0.01)^2,lty='dashed')
> lines(seq(-1,1,by=0.01),100*seq(-1,1,by=0.01)^2,lty='dotted')
> lines(seq(-1,1,by=0.01),abs(seq(-1,1,by=0.01))^1.25,col='red')
> lines(seq(-1,1,by=0.01),10*abs(seq(-1,1,by=0.01))^1.25,col='red',lty='dashed')
```

```
> lines(seq(-1,1,by=0.01),100*abs(seq(-1,1,by=0.01))^1.25,col='red',lty='dotted')
```

Table 1: Summary of convergence tests.

Optimization iterations	Nelder-Mead iterations	BFGS iterations	λ
20	2500	1000	1, halve for (2-20)
20	2500	1000	1, halve for (1-10), 0 for (11-20)
20	5000	1000	1, halve for (1-10), 0 for (11-20)
30	2500	1000	1, halve for (2-30)
50	2500	1000	1, halve for (2-50)

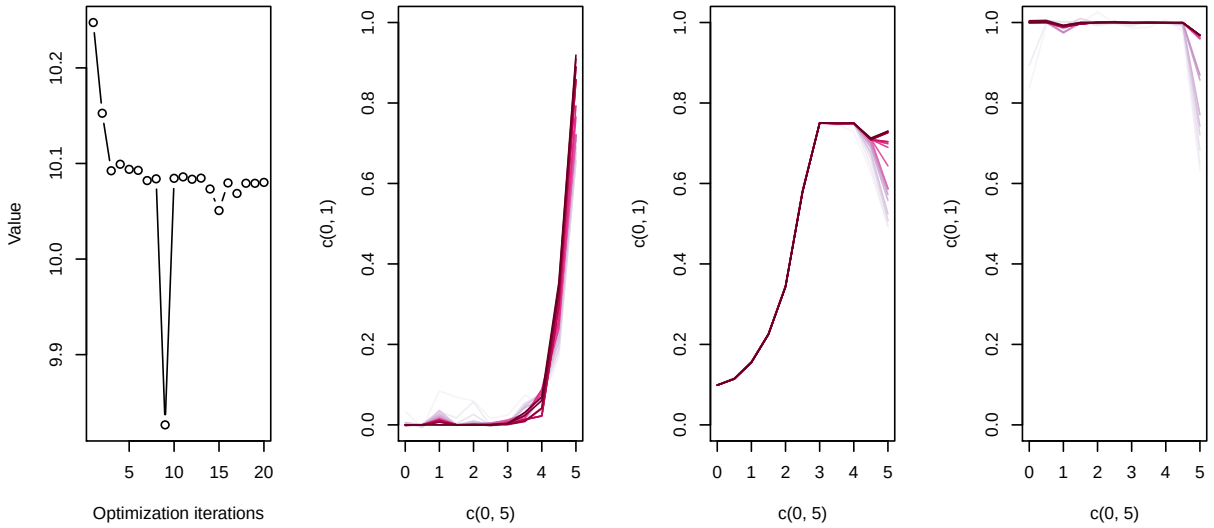

Figure 1: Optimization: 20 iterations, Nelder-Mead: 2500 iterations, BFGS: 1000 iterations, start lambda at 1, halve for 2-20

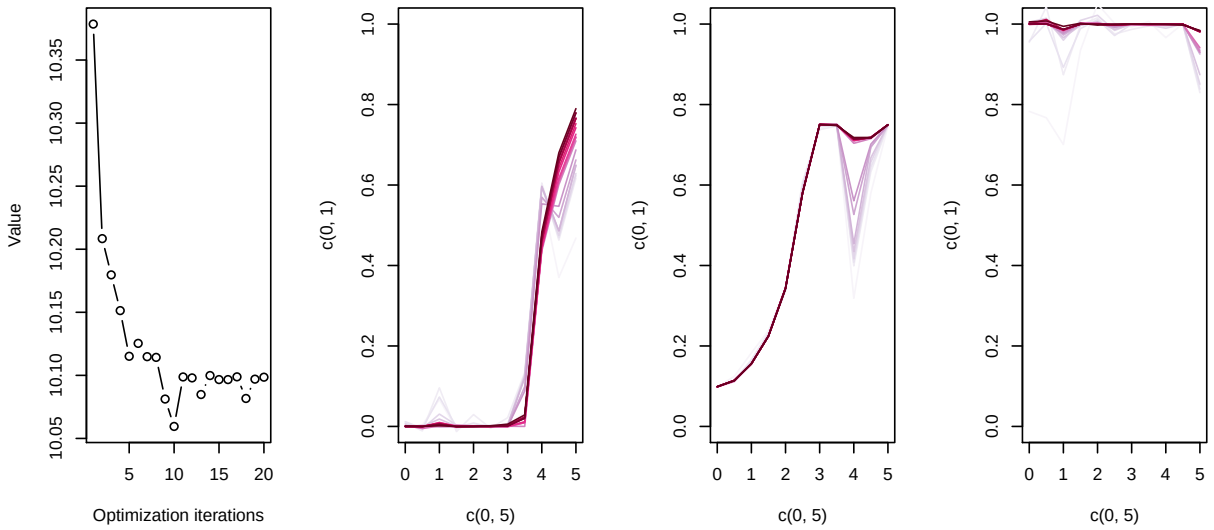

Figure 2: Optimization: 20 iterations, Nelder-Mead: 2500 iterations, BFGS: 1000 iterations, start lambda at 1, halve for (1-10), 0 for (11-20)

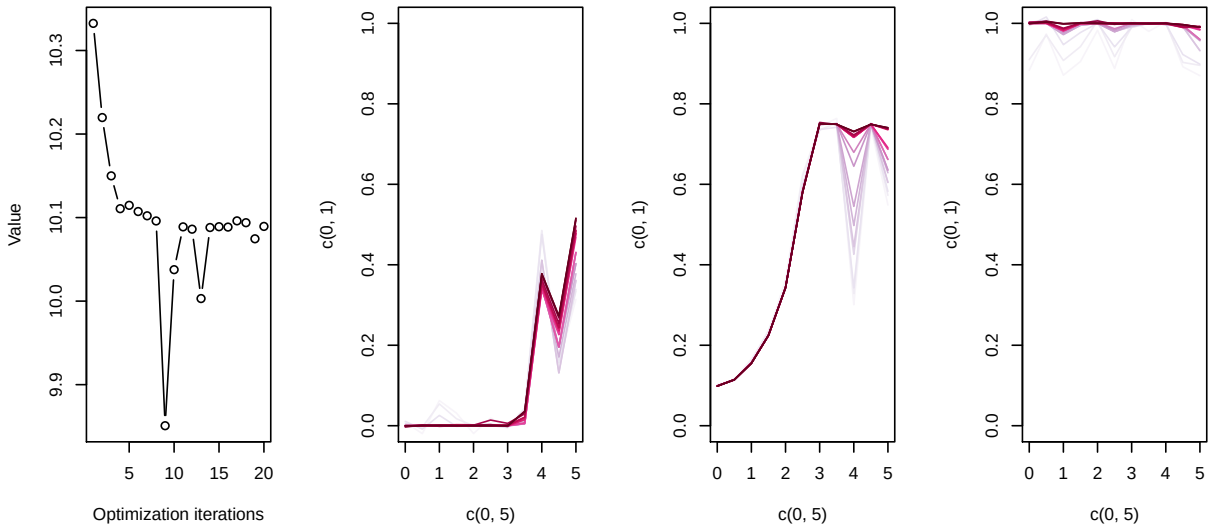

Figure 3: Optimization: 20 iterations, Nelder-Mead: 5000 iterations, BFGS: 1000 iterations, start lambda at 1, halve for (1-10), 0 for (11-20)

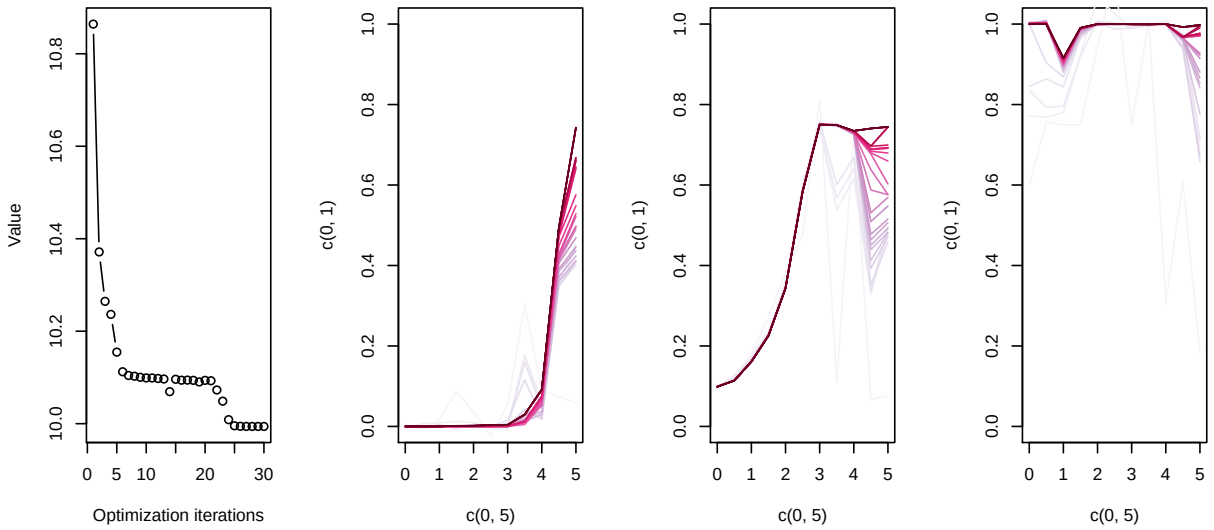

Figure 4: Optimization: 30 iterations, Nelder-Mead: 5000 iterations, BFGS: 1000 iterations, start lambda at 1, halve for (1-10), 0 for (11-30)

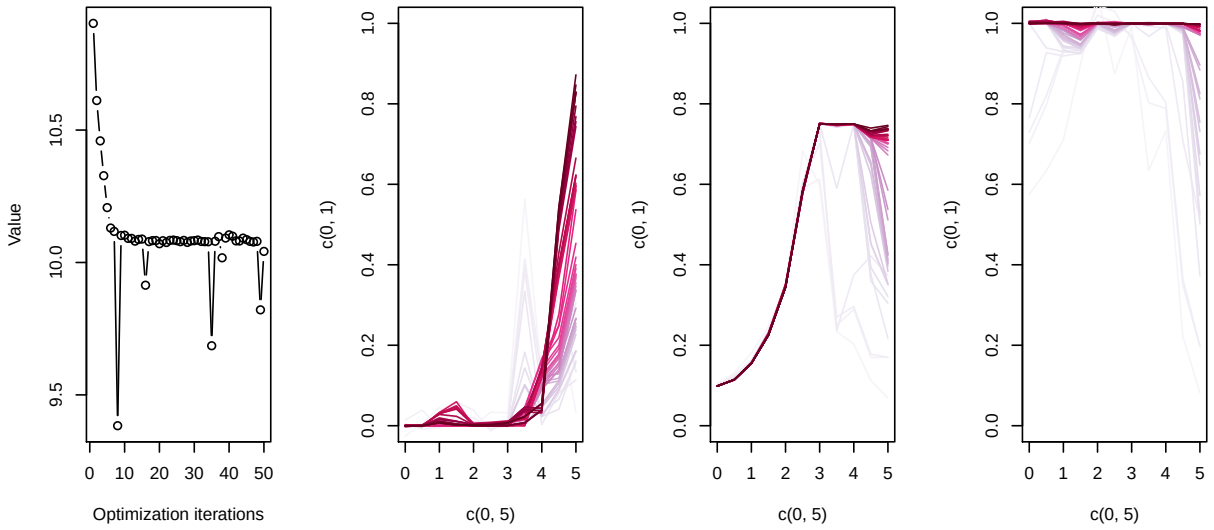

Figure 5: Optimization: 50 iterations, Nelder-Mead: 5000 iterations, BFGS: 1000 iterations, start lambda at 1, halve for (1-10), 0 for (11-50)