CAAM564 Final Project

Joey Lou

May 1, 2019

1 Introduction

As detailed in notes, the optimal control problem for steady state Burger's equation can be achieved through finite-element discretization. To briefly review the original problem:

$$min \quad \frac{1}{2}\int_0^1 (y(x)-\hat{y}(x))^2 + wu^2(x)dx,$$

$$s.t. \quad -v\frac{\partial^2 y(x)}{\partial x^2} + y(x)\frac{\partial y(x)}{\partial x} = r(x)+u(x) \quad x\in(0,1),$$
$$y(0)=y(1)=0,$$

where $y(x)$ is the steady state profile, $u(x)$ is the control, and $\hat{y}(x)$ is the profile we want to achieve. $y(x)$ and $u(x)$ are optimizing variables.

2 Methods

The infinite-dimensional problem was integrated and discretized using finite-element methods following the notes. Details will be spared in this report. Similarly, a finite-element numerical simulation was constructed to verify our optimized results. The set-up for the simulation is as follows:

$$\frac{\partial y(t,x)}{\partial t} - v\frac{\partial^2 y(t,x)}{\partial x^2} + y(t,x)\frac{\partial y(t,x)}{\partial x} = r(x)+u(x) \quad x\in(0,1),$$
$$y(t,0)=y(t,1)=0,\ y(0,x)=0.$$

3 Results

In our optimization problem, there are a few "fixed" variables that we can adjust to change the problem set-up.

- The diffusivity term v will change the dynamics. In general higher v results in larger forcing required to achieve target profile.
- Penalty term w on $u(x)$ will affect the optimal solution directly. Larger w results in smaller magnitudes in control and generally worse matching with target profile.
- Fixed forcing term $r(x)$ will change the control $u(x)$ directly.
- Target profile $\hat{y}(x)$ is chosen to be quadratic.

We fixed the aforementioned variables at: $v = 0.1$, $w = 0.01$, $r(x) = -0.1$, $\hat{y}(x) = (1 - x)x$.

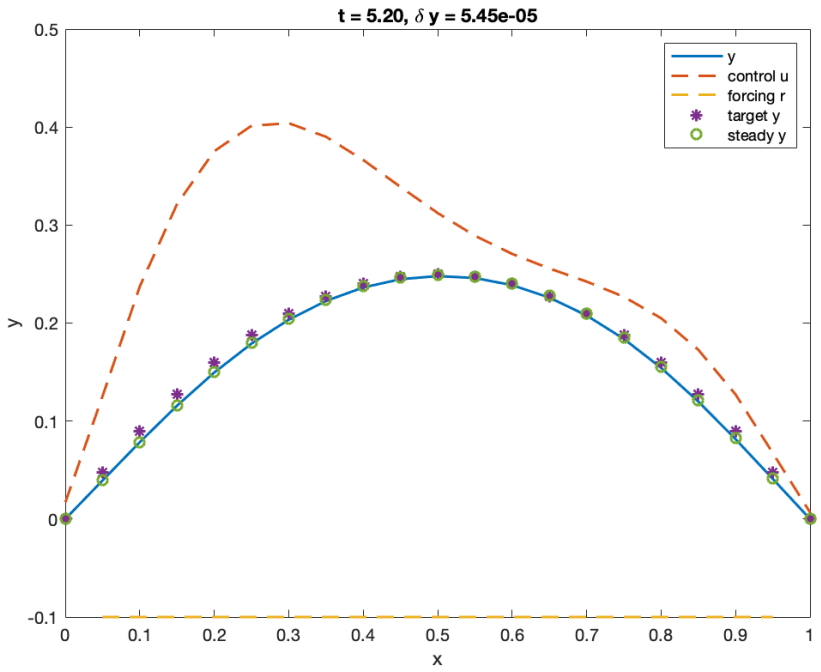

Figure 1: Sample Sim Run with Optimized Variables from SQP

To start optimizing $u(x), y(x)$, we first found strictly feasible starting points by running the numerical simulation with a given initial guess $u_0(x)$. Then, per the requirement of each algorithm, we provide necessary inputs such as Jacobian of objective and Hessian of Lagrangian.

We compared different Matlab optimization toolbox algorithms with the same set of conditions and initial guesses (except for run 6), and tabulated results below:

Run	Algorithm	Grad	Hess	Iteration	Objective value	Constraint violation	Final step size	First Order Optimality
1	SQP			48	-1.62E-02	3.87E-11	7.51E-04	3.56E-07
2	SQP	x		48	-1.62E-02	3.95E-11	7.65E-04	3.55E-07
3	Interior-point			20	-1.62E-02	3.24E-09	1.97E-03	3.59E-06
4	Interior-point	x		20	-1.62E-02	3.23E-09	1.97E-03	3.59E-06
5	Interior-point	x	x	400	-2.23E+304	2.22E+153	NaN	6.98E+149
6	Interior-point (start close to solution)	x	x	2	-1.62E-02	1.93E-07	1.41E-02	2.21E-07
7	Active-set			17	-1.62E-02	9.59E-07	8.55E-03	5.48E-05
8	Active-set	x		17	-1.62E-02	9.59E-07	8.55E-03	5.48E-05

where Grad refers to whether or not gradient information is provided and Hess refers to Hessian.

Here are some rather superficial discoveries from implementing this problem:

- By default, forward difference is used to approximate gradient, which is not far off from actual gradients as iteration counts did not change at all after analytical gradients are supplied for all three algorithms.
- Active-set performed the best out of the three algorithms from optimization toolbox on this problem. However, the optimality conditions were less desirable compared to the other two.
- There seems to be a trade-off between faster convergence and better constraint satisfaction among the three methods.
- Surprisingly, adding Hessian to interior-point method did not improve its performance. By default, BFGS is used to find approximated Hessian. Here, the exact Hessian only worked when initial guess was close to the solution.

All code is written in matlab and can be accessed from Github.