Report 03/09/2020

1 Last Week

Implemented the dot product in shape and mulitshape.

2 Tests

Tested the forward problem and optimal control problem in different settings. It all seems to be working very well. Forward Tests (Figure 1):

- Test 1: Compare computations on a box, with ADInf solution.
- Test 2: Compare on a box with AD Flow Neumann Exact solution.
- Test 3: Split box in MS code and compare to box in Box code using ADInf solution
- Test 3a: Same as 3 only checking that order of shapes don't matter.
- Test 4: Computing problem ADInf on wedge + quadrilateral
- Test 4a: Same as 4 only checking that order of shapes don't matter.
- ToyProblem 1: Computes no flux problem on two wedges and two quadrilaterals with constant 1 flow.

Optimization Tests (Figure 2):

- Test 5: Comparing MS and Box code on a box with Neumann Flow Exact Problem
- Test 6: Split MS box in two parts and compare to box in Box code (same Exact solution)
- Test 7: Comparing on the box an interacting problem (problem one from paper)
- Test 8: Splitting MS box and comparing to Box code for interacting problem

3 Comparing Matching Conditions in Forward problem

Compared with ADinf exact solution. Matching two boxes (vs full box) and matching two wedges (vs full wedge). Both show that the results are the same regardless of the matching method.
Using the example with no flux and two wedges and two boxes (see Figure 3), the two matching methods are compared. The error in ρ is 1.4292×10^{-10}.

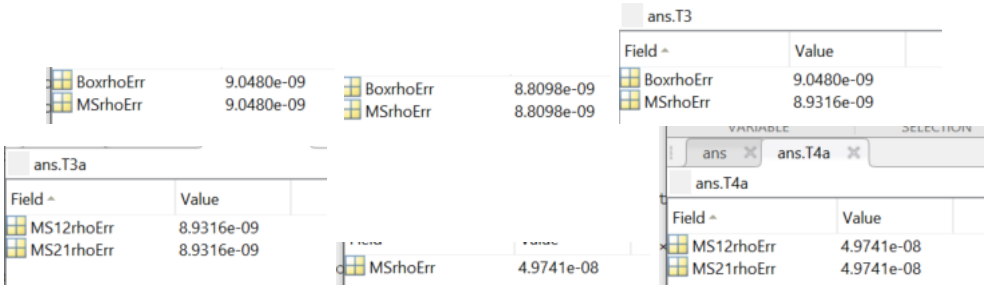

Figure 1: Forward Test Solutions

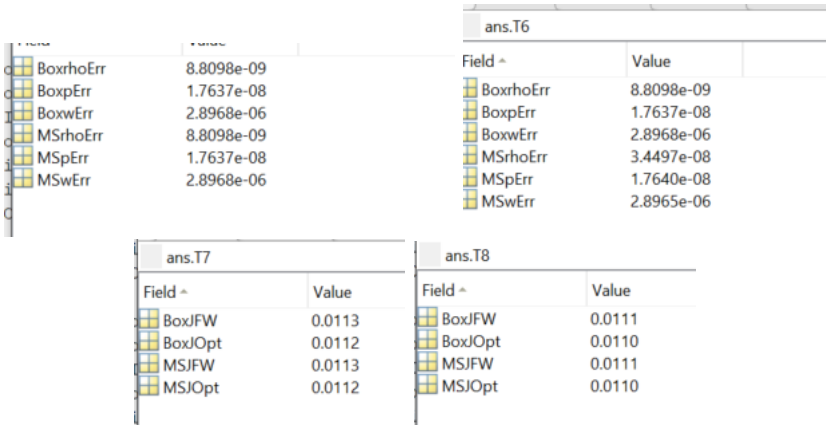

Figure 2: Optimization Test Solutions

4 Finding distances between points that leave the domain

See Figure 4. We decided that this is the way to do it. Still needs to be implemented. Aim: finding euclidean distance between each pair of points that crosses a boundary. This is to exclude those from the particle interaction, since those two particles then shouldn't interact.

5 OCP on MultiShape

5.1 Example 1

We choose the initial condition for ρ to be $exp(-2((y1-0.5)^2+(y2+0.5)^2))$ and solve a forward problem with constant velocity of strength one, see Figure 5. Then we use this forward solution as a target in the OCP, with initial velocity zero. We set $\beta = 10^{-3}$, we solve with tolerances $10^{-7}/10^{-3}$, because of time constraints, and $n = 20$, $N = 20$. The solution can be seen in Figure 6. As expected, the control follows the particle mass. It takes 452 iterations, but the time it takes is 2×10^4. We get $J_{FW} = 0.0206$, $J_{Opt} = 0.0020$. We then choose $\kappa = -1$ and get $J_{FW} = 0.0251$, $J_{Opt} = 0.0020$, in 454 iterations taking 1×10^4 in time. We can see the results in Figures 7 and 8. We can do the same for $\kappa = 1$. We get $J_{FW} = 0.0176$, $J_{Opt} = 0.0020$, in

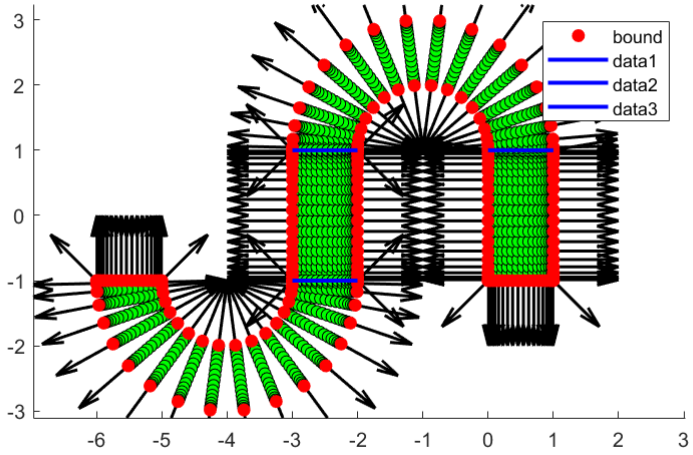

Figure 3: Domain

451 iterations taking 1×10^4 in time. We can see the results in Figures 9 and 10.

5.2 Example 2

We choose the initial condition for ρ to be $exp(-2((y1-0.5)^2+(y2+0.5)^2))$ and solve a forward problem with constant velocity of strength five, see Figure 11. Then we use this forward solution as a target in the OCP, with initial velocity zero. We set $\beta = 10^{-3}$, we solve with tolerances $10^{-7}/10^{-3}$, because of time constraints, and $n = 20$, $N = 20$. The solution can be seen in Figure 12. Again, as expected, the control follows the particle mass. It takes 587 iterations, but the time it takes is 5×10^4. We get $J_{FW} = 0.1921$, $J_{Opt} = 0.0326$. Figures 13 and 14 show the results for different interactions.

6 Things that do not work

1. Giving the forward problem with the constant velocity as initial guess AND as target for the OCP and ask it to do better. It immediately diverges. Maybe too advection dominant?
2. Considering problem 2 above with velocity strength 10 and interaction term. Diverges. Advection dominance?
3. In and outflow BCs. Diverges immediately. Correct implementation? Or maybe this restricts too much and we can't actually change the flow much...

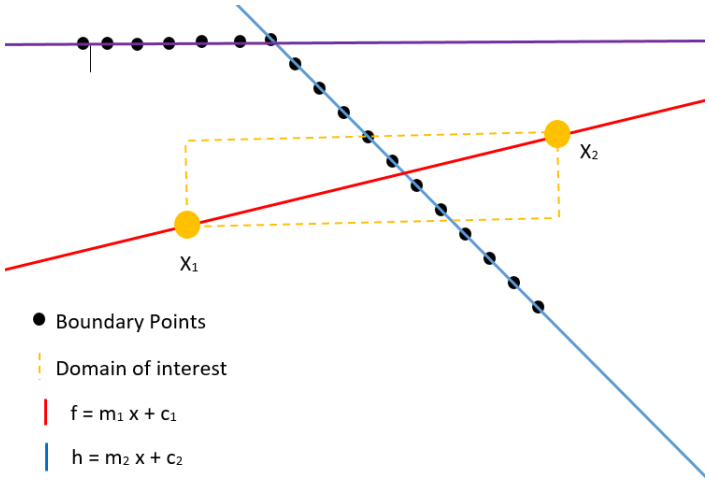

Figure 4: Intersection

7 This Week

Regarding the things that do not work: 1. Probably a bad initial guess. 2. Instead of higher velocity, we run the problem for longer times. This seems to work (at least for the first few iteration) but it's running on the server at the moment/ the server is down so it might not be. Each iteration takes appropriately longer. 3. We said that $w = 0$ may not be a good initial guess, but $w = 1$ may be. It isn't it diverges too very quickly. We also said that this is probably a very hard OCP to solve (even without MultiShape), so we need more investigation there – maybe on a box. Ben said to park this problem for now. However, what about source control or inflow control. Is it worth writing down an inflow control problem maybe?

8 Time

The problems on multishape take more time than the problems without. Tested Advection-Diffusion Flow Control Exact problem on a box with $N = 30$, $n = 20$. This takes 36 seconds with multishape and 9 seconds for the standard box. Maybe I am measuring wrong but I don't think so. When doing the same with $N = 20$, $n = 20$ we get that the MultiShape box solves in 4 seconds and the standard Box in 1.6 seconds. When splitting the box into two parts and evaluate each 'subbox' with $N = 20$, $n = 20$, this solves in 25 seconds. This is faster than the 'whole' box with $N = 30$, which I think is slightly odd. But not sure.

I incorporated a count in the code now to evaluate how long each iteration takes (instead of just total time). Since then I haven't run a long OCP to the end, but the first few hundred iterations take each roughly the same amount of time. For the two problems above this is on

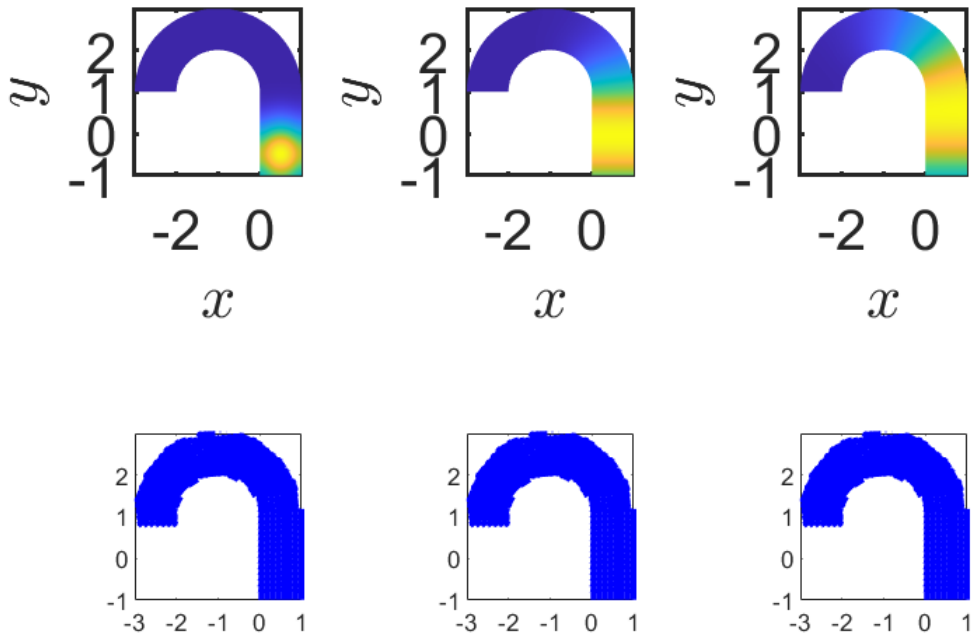

Figure 5: Test1 Forward $t = 1, 10, 19, n = 20$

average 20 seconds per iteration. The fancy channel (see Figure 15 for target) however takes 180 seconds per iteration, which is why I had to stop it on my machine after 250 iterations. It converged to almost 10^{-2} in that time though so I am reasonably confident that this works. The target is a background flow of strength one, and diffusion.

9 Time savings

We talked about a few things that I can do to make the code quicker. The first one is to solve a problem on a sparse grid, interpolate the solution onto a fine grid and run it again, hoping the IG from the sparse grid will make convergence faster. This will probably work, but I am currently stuck at the interpolation matrix. It is 'loosing points'.
The second idea is to take the solution for $\kappa = 0$ to solve for $\kappa = 1$, $\kappa = -1$. This is because the solutions for the three interaction strengths are not as different from each other as they are from $w = 0$. I have tried this for $\kappa = 1$. This converges in 347 iteration, which is a saving of 104 iterations compared to starting with $w = 0$. Each iteration takes around 20 seconds, but

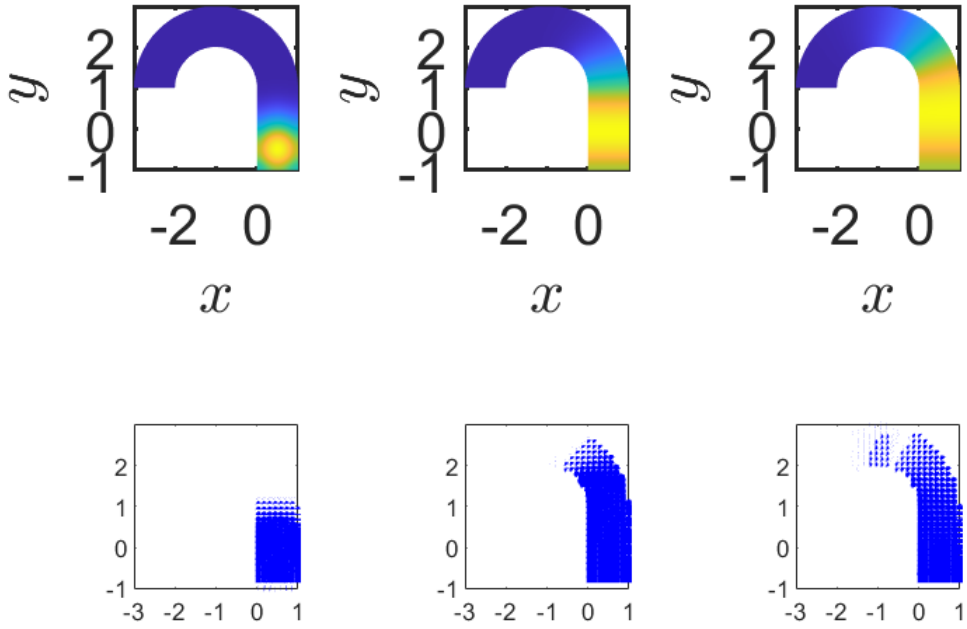

Figure 6: Test1 Optimization $t = 1, 10, 19, n = 20$

since I was doing other things on my laptop, resulting in iterations that took much longer, this averages to 22.5 seconds per iteration.
Another thing to do is to datastore the multishape itself but I have issues with this at the moment.
Another thing is to do an adaptive λ, but haven't tried that yet because if you get it wrong, you need to start at the beginning and that takes hours.

10 Other

I had some issues with matlab having memory issues and it is showing on the task manager too that at times it uses literally all of my laptops memory. Why? A lot of recovering results from datastorage? What can be done?

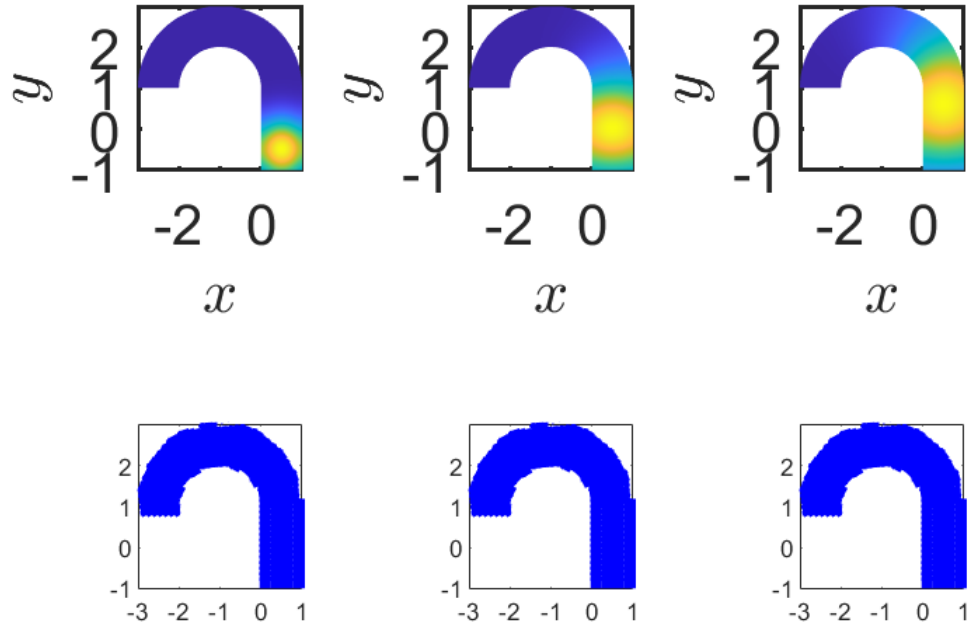

Figure 7: Test1 Forward $\kappa = -1$, $t = 1, 10, 19, n = 20$

11 Classes

How to enrol in Introduction to Optimization as listener only.
How do I enrol in School of Informatics classes?
Do we know when the Python class starts?
Should I take the MAC-MIGS course for credits or something else (given that we said I should get my credits done in semester 1).

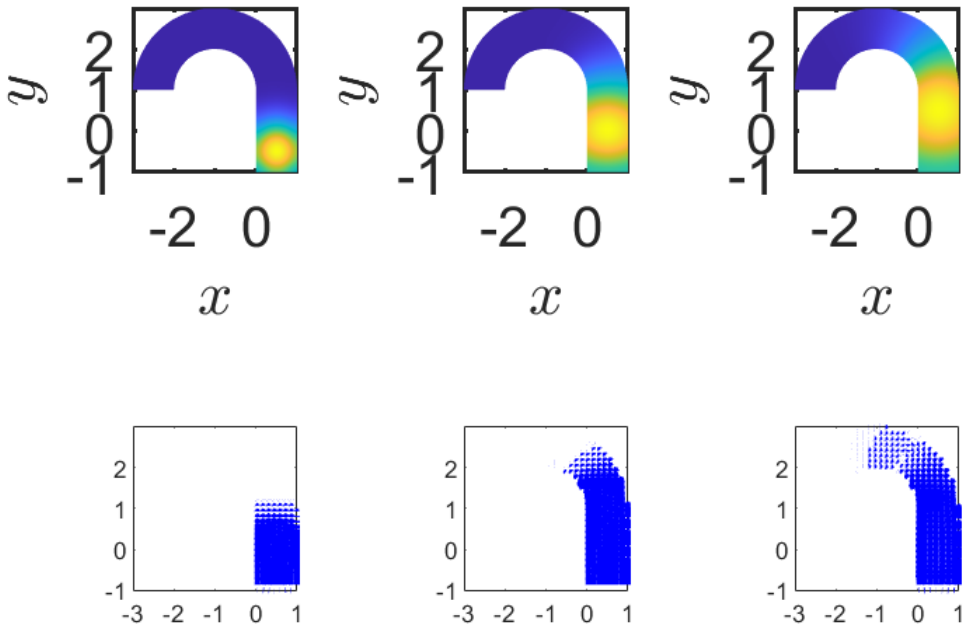

Figure 8: Test1 Optimization $\kappa = -1$, $t = 1, 10, 19, n = 20$

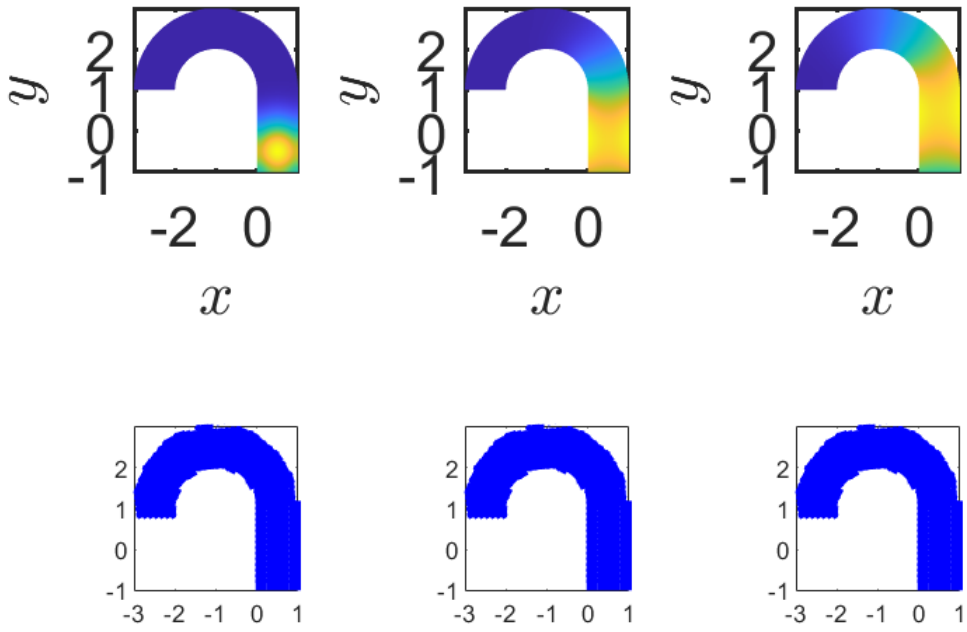

Figure 9: Test1 Forward $\kappa = 1$, $t = 1, 10, 19, n = 20$

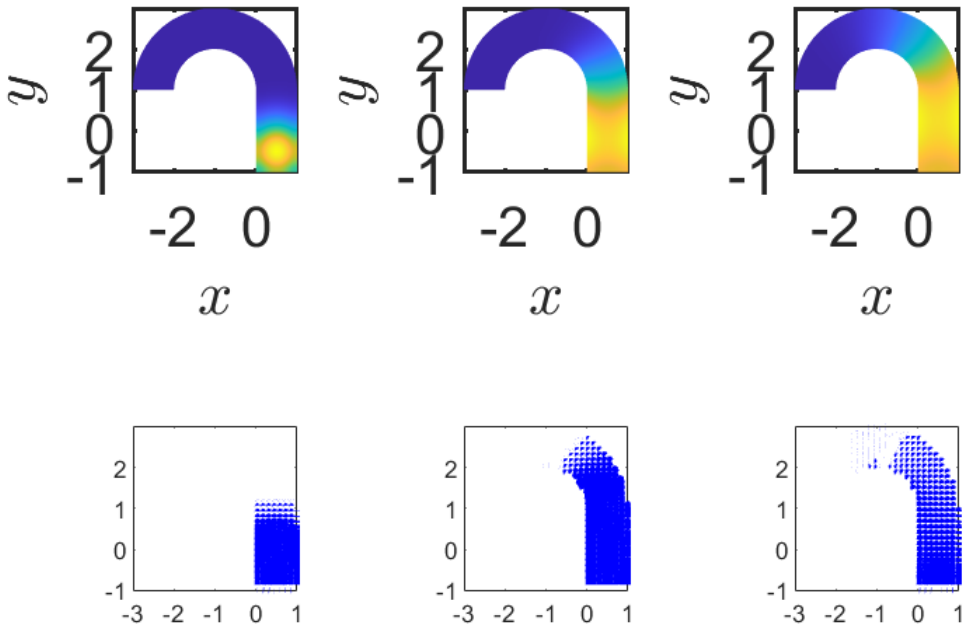

Figure 10: Test1 Optimization $\kappa = 1$ $t = 1, 10, 19, n = 20$

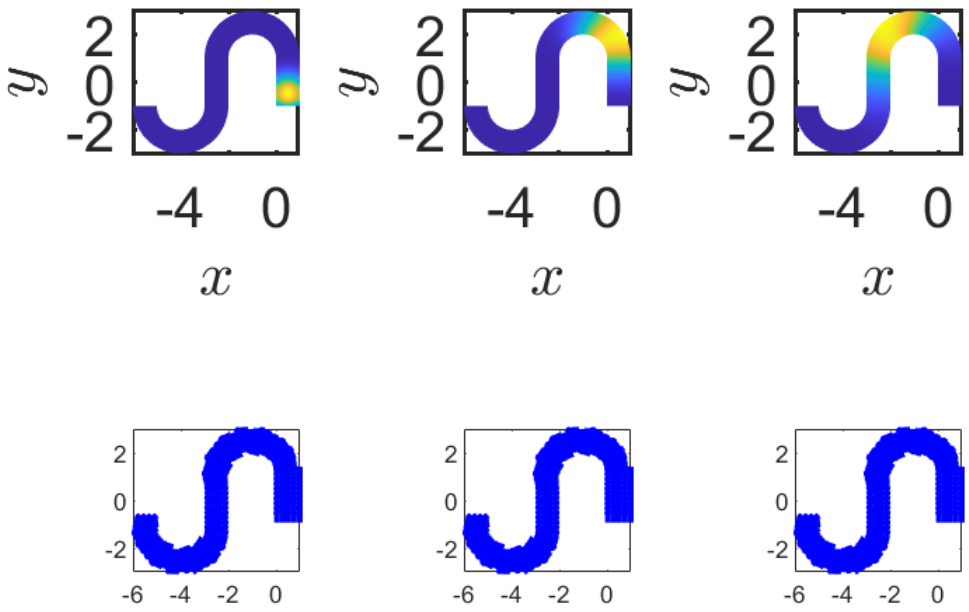

Figure 11: Test2 Forward $t = 1, 10, 19, n = 20$

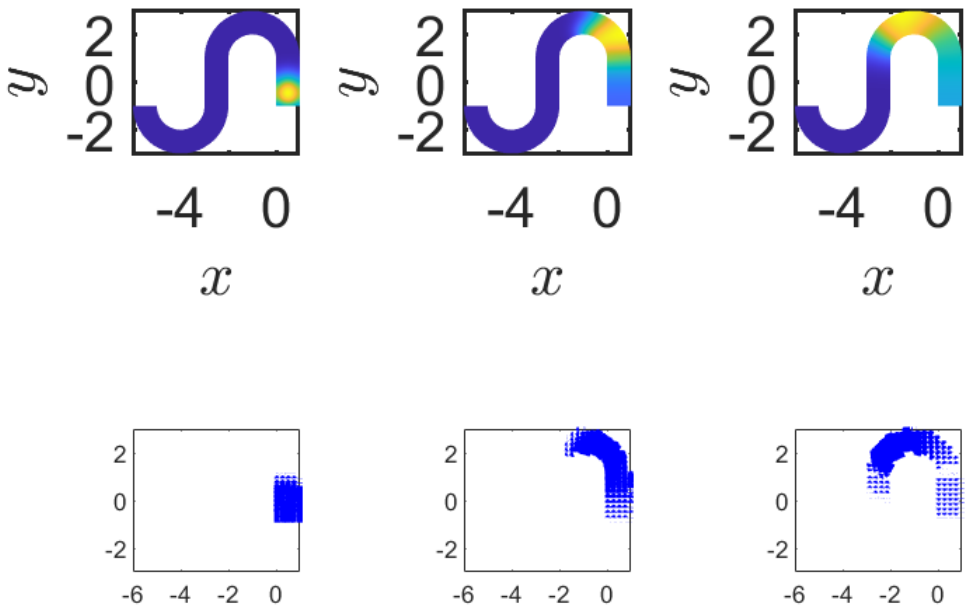

Figure 12: Test2 Optimization $\kappa = 0$, $t = 1, 10, 19, n = 20$

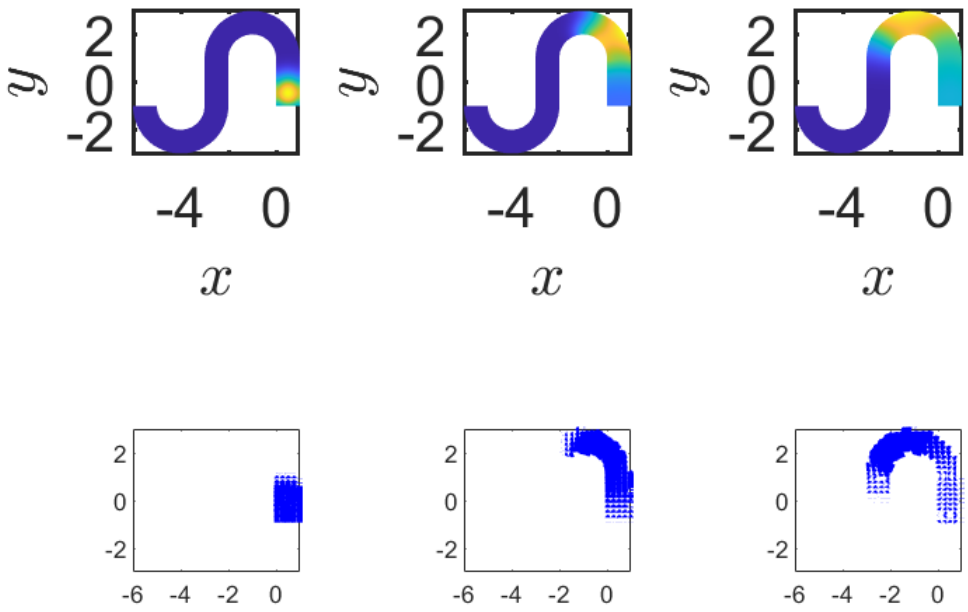

Figure 13: Test2 Optimization $\kappa = 1$, $t = 1, 10, 19, n = 20$

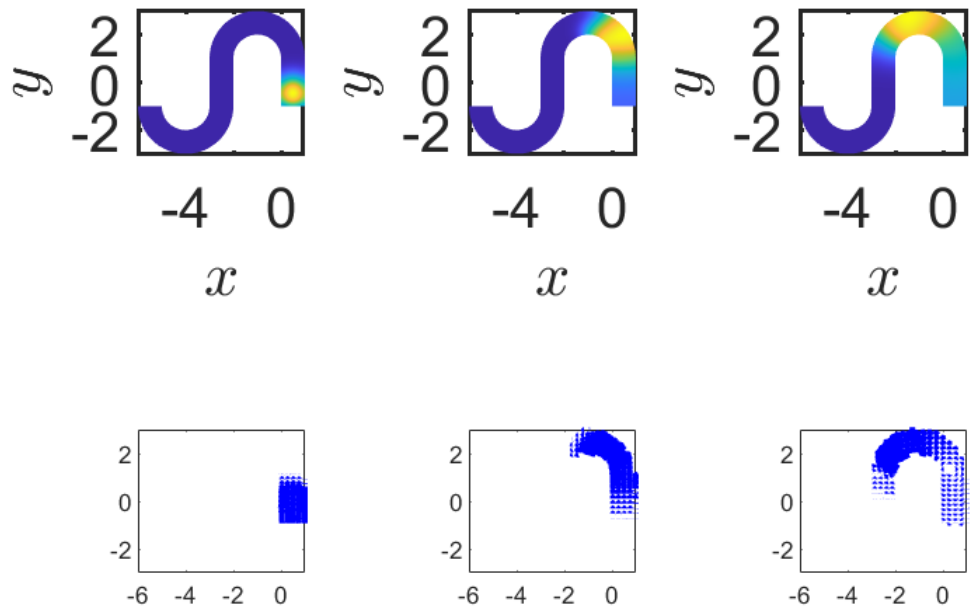

Figure 14: Test2 Optimization $\kappa = -1$, $t = 1, 10, 19, n = 20$

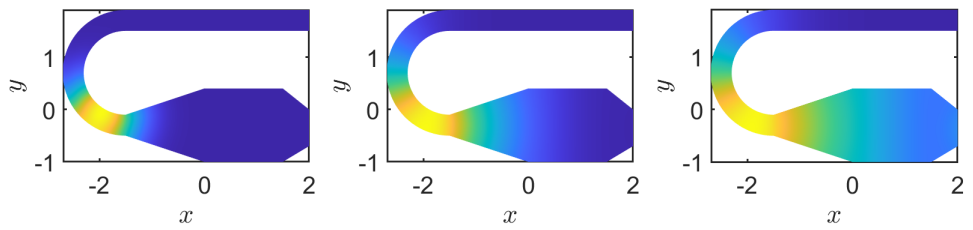

Figure 15: Fancy Channel Target