Nathaniel Rogalskyj October 27, 2016

Math 6230 Problem Number 3

Suppose that you want to walk from (-2,0) starting at time 0 and end at (0,-2) at time π while maximizing average cell phone reception, where the cell phone reception is given as a function of position:

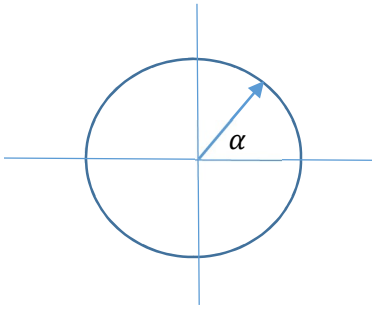
$$Q(x,y) = \begin{matrix} 20 - 5x^2 - 5y^2 \; if \; x^2 + y^2 < 4 \\ 0 \; if \; x^2 + y^2 \geq 4 \end{matrix}$$

I took a Calculus of Variations approach to the problem for a few reasons: first, I have a good initial guess for velocities $v = \frac{2\sqrt{2}}{\pi}$ and $v = \frac{4}{\pi}$. The first guess is a straight line from (-2,0) to (0,-2). The second guess is non-differentiable path from (-2,0) to the origin (0,0) and then from the origin (0,0) to (0,-2).

Non-Linear Optimization seems difficult since it would be difficult to incorporate the fact that an earlier control choice affects a later control.

The Dynamic Programming method would require extra computation since I would be in the area of closed loop control and I would be required to calculate the value function at all possible points in the domain, unless I had some ability to dynamically update what would be a viable section of the domain to use and this seems difficult.

We formulate the problem as a control problem where the control represents the angle of direction of travel made with the x-axis α: $[0, \pi] \rightarrow [0, 2\pi]$ as shown below:



The dynamics are then given by

$$\dot{x}(t) = \begin{bmatrix} vcos(\alpha) \\ vsin(\alpha) \end{bmatrix}$$

$$x(0) = \begin{bmatrix} -2 \\ 0 \end{bmatrix}$$

$$P(\alpha) = \int_0^{\pi} Q\big(x(t)\big)dt$$

Looking at Evan's Control Theory Notes for the fixed time problem, I apply the Pontraygin Maximum Principle to get that $\alpha^*(t)$ and $x^*(t)$ satisfy:

$$H(x,p,\alpha) = f(x,\alpha)p + r(x,\alpha)$$

$$\dot{x}^*(\mathrm{t}) = -\nabla_p H(x^*(t), p^*(t), \alpha^*(t))$$

$$\dot{p}^*(\mathrm{t}) = -\nabla_x H(x^*(t), p^*(t), \alpha^*(t))$$

$$H\big(x^*(t), p^*(t), \alpha^*(t)\big) = \max_{\alpha \in A} H(x^*(t), p^*(t), \alpha)$$

With a change of notation I let x(1)=x and x(2)=y. I also set the first component of p to be p_1 and the second component of p to be p_2. So I get for the Hamiltonian the following:

$$H(x,p,\alpha) = vcos(\alpha)p_1 + vsin(\alpha)p_2 + 20 - 5x^2 - 5y^2$$

Using the statement that

$$H\big(x^*(t), p^*(t), \alpha^*(t)\big) = \max_{\alpha \in A} H(x^*(t), p^*(t), \alpha)$$

We can minimize with respect to α, and we get that

$$\alpha = Arctan(\frac{p_2}{p_1})$$

Substituting back into the differential equations provided by the statement of Pontraygin Maximum

$$\dot{x}^*(\mathrm{t}) = \mathrm{v} * \cos(Arctan\left(\frac{p_2}{p_1}\right))$$

$$\dot{y}^*(\mathrm{t}) = \mathrm{v} * \sin(Arctan\left(\frac{p_2}{p_1}\right))$$

$$\dot{p_1}^*(\mathrm{t}) = -10\mathrm{x}$$

$$\dot{p_2}^*(\mathrm{t}) = -10\mathrm{y}$$

$$x(0) = -2$$

$$y(0) = 0$$

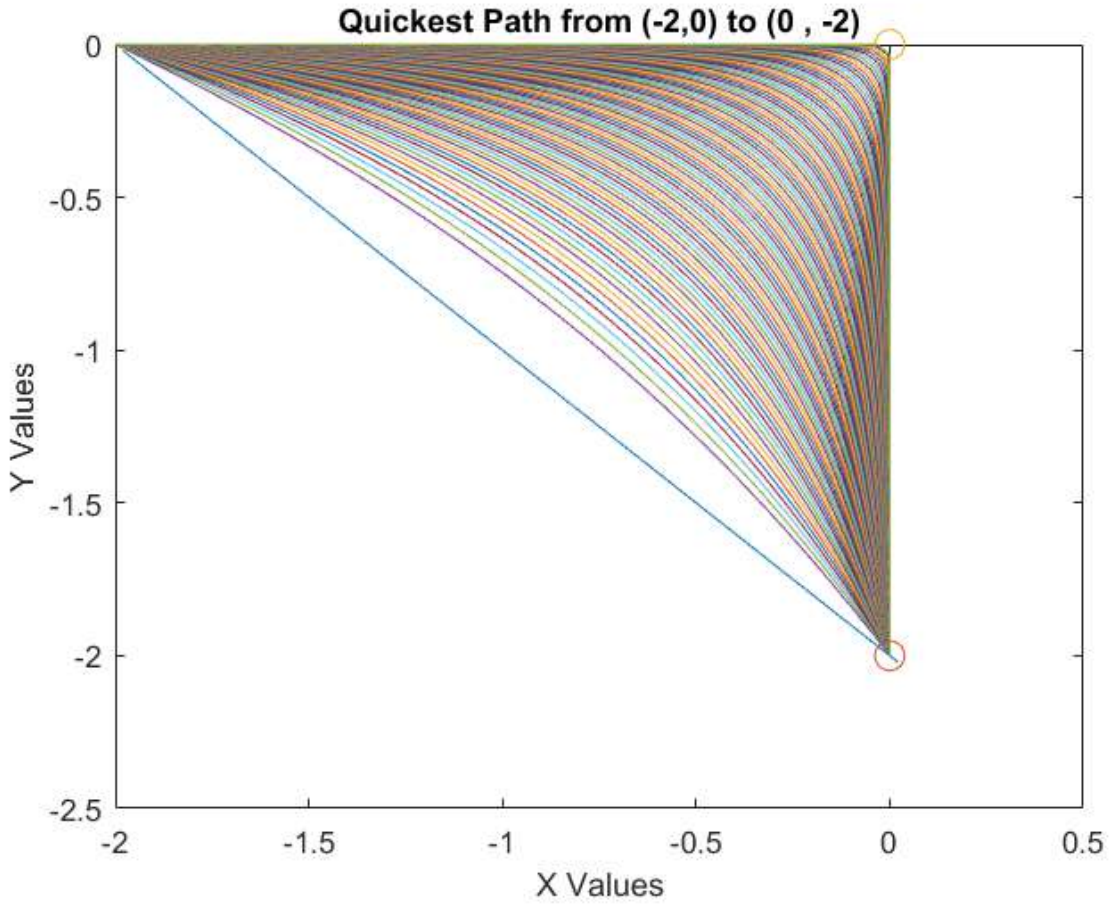
Since I also know the terminal points, I also set

$$x(\pi) = 0$$

$$y(\pi) = -2$$

The above sets up a Boundary Value Problem. At first trying to solve the above with a guess of a line for the values of x and y with $p_1 = 0$ and $p_2 = 0$ did not work and gave me the error that my Jacobian was singular. In order to fix the problem I solved for the corresponding p_1 and p_2 for the straight line solution with v=$\frac{2\sqrt{2}}{\pi}$ analytically. This worked well. And gave me good solutions for v close to $\frac{2\sqrt{2}}{\pi}$. However it was difficult to use this guess to converge to higher values of v. I thereby employed

continuation method where I used a previous solution to a lower v to be the guess to a bvp with a higher v. The results of this are provided below.



When v is greater than $\frac{4}{\pi}$, the person will move as quickly as possible to the origin sit there and then move to (0,-2). For v less than $\frac{2\sqrt{2}}{\pi}$ the person will not be able to make it to the ending point in time and no solution exists.

The method becomes more accurate by imposing a finer mesh of time values. The method employed by bvp4c in this example is the collocation method, where the solution is taken as a linear combination of basis functions. The accuracy of the solution is determined by the smoothness of the function. Clearly for more discontinuous paths such as those with $\frac{4}{\pi}$, the solution error will be much greater as the function is no longer smooth. Hence we expect to have to use basis function with local support for large v and global basis functions for small v. The computational cost of converging to the correct answer scales by the number of points used. The error goes down linearly in the number of points. This is worse than the dynamic programming solution since to get a 2 fold increase in accuracy in one direction, one needs 4 more grid points on the domain. Expect that to scale proportional to $\sqrt{n}$.

Another concern is whether the guess is close enough to the correct local maximum in order to build up to a hard to converge problem. As I experimented a fixed number of 100 steps is good enough to force convergence throughout the entire velocity range.

```
%Solver with Continuation


%Parameters
velocity_interval = [0.91 , 1.273] ;
number_steps = 100;
velocity_step_size = (velocity_interval(2)-
velocity_interval(1))/number_steps;
%End Parameters%


velocity = velocity_interval(1);


starting_point = [-2 ; 0];
closeness_bound = 1;
%Higher values more accuracy in computed solution
tightness_bound = 1;


yinit = @(x) guess_function( x , starting_point, velocity
);
%Guess is straight line from (-2, 0) to (0, -2) at velocity
v.
%Plot Guess Solution
start_time = 0;
end_time = pi;
Number_time_increments = 100;
time_step = (start_time- end_time) /
Number_time_increments;
accumulation = [];
for i = 1 : Number_time_increments
    time = i * time_step ;
    row_vector = yinit(time);
    accumulation = [accumulation ; row_vector];
end

plot (accumulation(:, 1), accumulation(:, 2)); hold on
plot(0,-2,'o', 'markersize', 10) ; hold on
plot (0 , 0 , 'o', 'markersize', 10);
x = [0 , pi];


sol= bvpinit(x, yinit);
```

```
for i = 1 : number_steps
    display(velocity);



    odefun = @(x,y) [velocity*cos( atan ( y(4) / y(3))) ,
velocity*sin( atan ( y(4) / y(3))) ,  -10*y(1) , -10*y(2)];
    mesh_size = 100;
    bcfun = @(ya , yb) eikBC(ya,yb);
    options = bvpset('NMax',mesh_size, 'RelTol', 1e-13,
'AbsTol', 1e-13, 'Stats', 'on' , 'BCJacobian',@eikBCJac);
    sol = bvp4c(odefun,bcfun,sol, options) ;


    solution_values = sol.y;
    %display(solution_values);
    x_vals = solution_values(1, :);
    y_vals = solution_values(2, :);


    %plot the solution

    plot (x_vals, y_vals); hold on

    velocity = velocity + velocity_step_size;


end
```