# 4M20 Robotics - Investigation into a Planar 3-link Manipulator

**Candidate number: 5614A**

Fully-actuated 3-link manipulators are reasonably common in nature, with most animal arms and legs exhibiting this rough design. The following is a theoretical investigation into a planar 3-link manipulator, although the manipulator considered here is far more constrained than most biological limbs, which often have more degrees of freedom at each joint, but also have to act in a three dimensional world. For the majority of tasks, the design discussed here is kinematically redundant - the manipulator has more degrees of freedom than required for the simple task of moving the end effector to a desired position. This redundancy gives extra dexterity, particularly around obstacles, and reduces the joint torques and velocities required for most tasks. Limiting it to planar motion makes calculations significantly easier. One could generalise this design to a 3 dimensional manipulator by adding a rotational joint at the base.

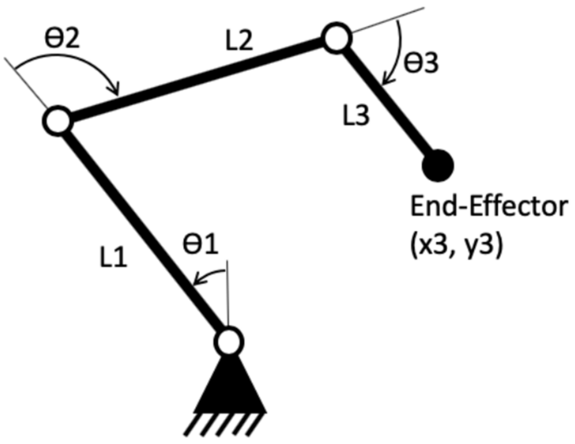


*Figure 1: Fully-actuated 3-link manipulator considered in this report*

## 1 Control and Planning

### 1.1 Kinematic Control

Kinematic control uses kinematics - the positions and velocities of actuators, without considering forces, torques or accelerations - to control motion. This is a major simplification, but allows much more efficient computation of motion control. Two more concepts are key - forward kinematics finds the overall robot position, given the intended joint angles, while inverse kinematics finds the joint angles required to enact a given overall robot position.

In the case of the manipulator considered here, the overall robot position refers to the position of the end effector. In kinematically redundant manipulators there can be many solutions to a given end effector position $(x_3, y_3)$. The robot can also have position constrained to a subset of the space this space it can move within is known as the reachable workspace or range.

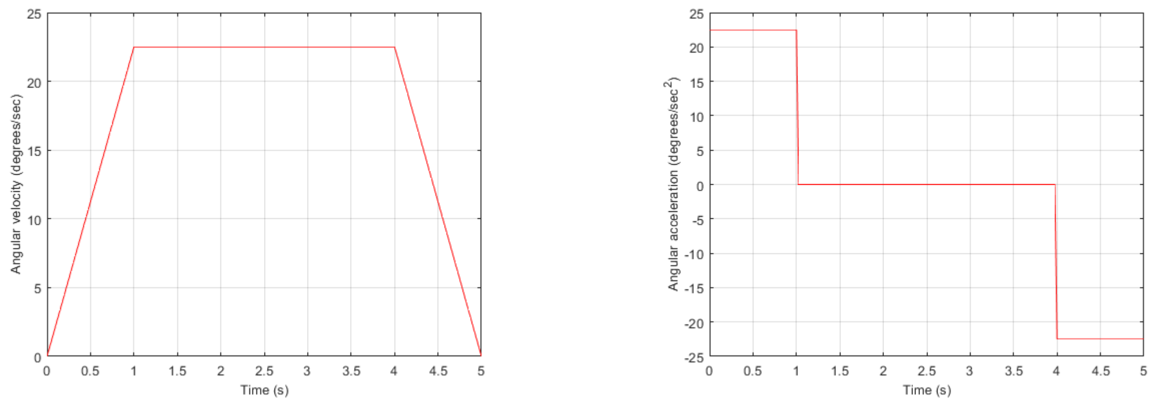
### 1.2 Dynamic Control

Dynamic control pertains to the dynamic equations of motion of a robot, and necessarily includes all forces acting on a robot. Similar to with kinematics, dynamics has its own forward and inverse algorithms. Forward dynamics determines the trajectory of the robot given joint forces and torques over time, while inverse dynamics determines joint forces and torques over time given the intended robot trajectory. While the computational requirement goes up when moving from kinematic control to dynamic control, so does the accuracy of the motion predicted. This is because dynamic control can also take into account other effects, such as motor slip, torque limits and joint angular velocity limits.

## 1.3 Planning Algorithms

While control methods, kinematic or dynamic, can find the set of joint movements or forces/torques required to achieve a given trajectory, there will often be an infinite number of possible trajectories to move from point A to B. Planning algorithms determine a trajectory, given a set of constraints. For the robot were considering, many different constraints and objective functions could be considered. Constraints could be any of: the start and end positions and velocities end-effector, the start or end angles/angular velocities of joints, maximum joint angular velocities or torques. Minimising the maximum joint angular velocity, mean joint angular velocity, or even the mean angular accelerations, could be objective functions.

There is also the inevitable tradeoff between optimality in the trajectory found and computation required. The balance of this tradeoff is typically set by the constraints on a system. If a system has a long time to plan its motion and therefore can afford use up far more computation, then the algorithm can find a more optimal solution. However, the majority of systems must plan under the constraint of finite computation, so finding a good solution quickly can be far more valuable than finding the perfect solution with far higher computational requirements.



*a)* *b)*

*Figure 2: Example of trapezoidal angular velocity and acceleration profiles*

One set of planning algorithms uses third (or higher) order polynomials to define robotic trajectories. The constraints are typically starting joint positions and velocities and the final robot position and velocity. These methods can guarantee smoothness in any time-derivative of position, $x$, with a high degree polynomial (ie can guarantee smooth $\dot{x}$ or $\ddot{x}$). However polynomial approaches are rarely used practically, so aren't considered much in this report. Instead, trapezoidal velocity profiles, such as figure 2a), are often used as they have constant torque steps as acceleration profiles, as can be seen in figure 2b), which are easily realisable with motors.

Converting to discrete planning problems can simplify them as 'no geometric models or differential equations will be needed to characterize discrete planning problems' - LaValle (2006). Typically optimality is inessential and a good solution with minimal computation is preferable. In this case discretisation is a good idea as it reduces computation and makes the problem tractable, but may not find the optimal solution.

## 2 Catching a Ball

Using the tools outlined above, the problem of catching a ball with this robotic manipulator can be tackled. The ball starts at point $(X_A, 0)$, is thrown at angle $\alpha$ at velocity $V_0$. The robot specifications are as follows: $L_1 = 0.8m$, $L_2 = 0.8m$, $L_3 = 0.3m$. The movable range of each joint is $\theta_1 = [-90°, +90°]$, $\theta_2 = [-160°, +160°]$, $\theta_3 = [-160°, +160°]$.

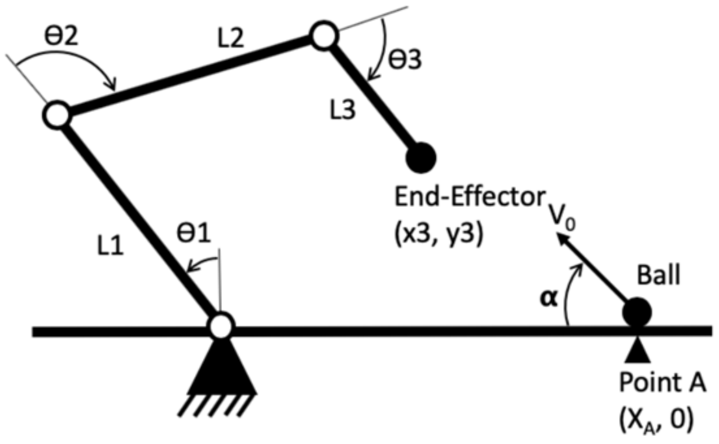


*Figure 3: Diagram of the ball-catching problem*

### 2.1 The Reachable Range

The problem of catching a ball can only be addresed once the the reachable range, or workspace of this manipulator is found. This range is the red region in Figure 2.1.

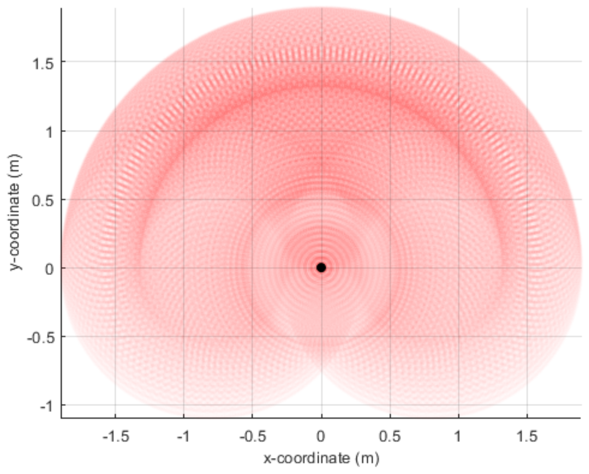


*Figure 4: Reachable range of fully-actuated 3-link manipulator*

The range is composed of the union of 3 smaller shapes. The first shape is a semicircle with radius 1.9m which is centred on the origin and is oriented such that it is entirely within the positive-y half-plane. This is drawn out when $\theta_2 = \theta_3 = 0$ for $\theta_1 = [-90°, +90°]$. The other two shapes are circles of radius $1.1m$ and are centred on the points $(0.8, 0)$ and $(-0.8, 0)$. These are drawn out when $\theta_1 \in \{-90°, +90°\}$ and $\theta_3 = 0$ for variable $\theta_2$.

If setting the position of the end effector, (x3, y3), with no obstacles present, the 3-link manipulator has more degrees of freedom, 3, than constraints, 2. As there are fewer constraints than degrees of freedom: for all interior points of the region plotted, there are multiple combinations of joint angles that can achieve that end-effector position. However, for points on the outside of the region, only one combination of joint angles can achieve it, as the manipulator cannot stretch out any further.

These plots were made by looking at 200 joint angles for each joint at regular intervals spread across the range of allowed joint angles. The end-effector position was then plotted as points with some transparency at all combinations of these joint angles. This means the level of shading in the reachable range shows roughly how many combinations of joint angles can reach those end-effector positions. Thus you can clearly see that the negative-$y$ half-plane have less redundancy, as $\theta_1$ is near, or at, the edge of its joint range so the manipulator is effectively a 2-link manipulator reaching below the $x$-axis.

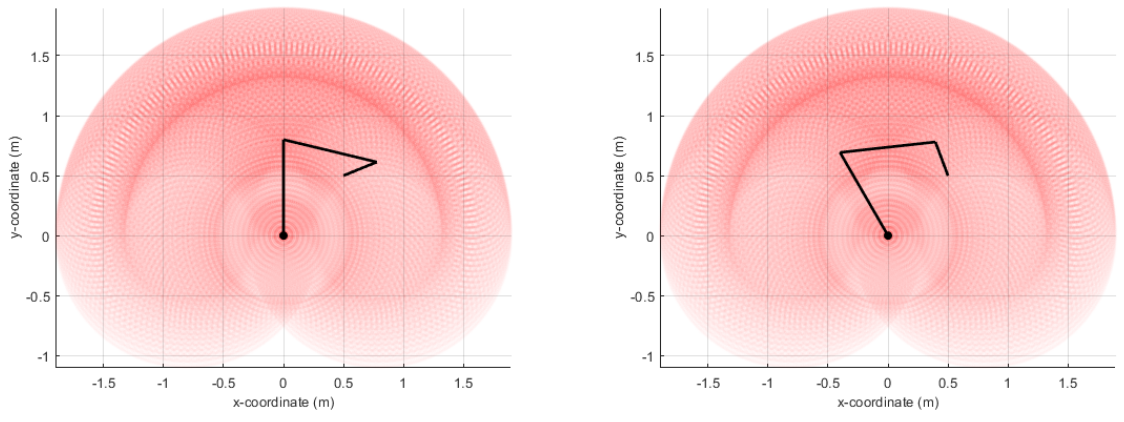
## 2.2 Positioning the End Effector

Given the end-effector position of $(0.5, 0.5)$, inverse kinematics can find possible joint angles. As there can be multiple combinations of joint angles that give the required end-effector position, it seems sensible to try to quantify whether some joint angles are better than others. Some heuristics for comparison are explained below.

The first way to compare sets of joint angles is the $l_\infty$-norm of the vector, $\delta\boldsymbol{\theta}$, where $\delta\boldsymbol{\theta}$ is the vector of changes in each joint angle. Minimising this is typically worthwhile if speed is necessary as in practice, joint actuators often have a maximum angular velocity. If this is the case, the largest change in joint angle constrains the time it takes to reach the new position. Or, one could compare the $l_1$-norm of $\delta\boldsymbol{\theta}$: the sum of joint angle changes. This is one way to minimise energy usage - a shorter joint angle change means lower average joint angular velocities to reach a given point in a set time.

Looking at the final position is a final way to assess how good a set of joint angles are. The closer each angle is to 0 at the final point, the better the point is. This is because, assuming the manipulator will move again, the next movement is more likely to have a lower $\delta\boldsymbol{\theta}$ across both $l_1$ and $l_\infty$ norms, as the vector $\mathbf{0}$ is the centre the joint angle range for every joint. This way of assessing a final position is usually less important than the other two mentioned. However no starting position is given for this problem, so it is the only way to compare possible sets of joint angles.

Figure 2.2 shows two possible sets of joint angles: 2.2a) has $\boldsymbol{\theta}= (0.0°, 103.5°, 144.3°)^T$, while 2.2b) has $\boldsymbol{\theta}= (30.0°, 113.7°, 75.8°)^T$. Using the third judgement criterion defined above, 2.2b) is superior as the maximum joint angle is smaller, and the sum of the joint angles is smaller, meaning the expected change in joint angles in subsequent motion is smaller. Optimal $\boldsymbol{\theta}$ wasn't found as no objective function was given to optimise over.



*a)* *b)*

*Figure 5: Two sets of joint angles with end-effector position* (0.5, 0.5)

## 2.3 Ball Catching ($\alpha = 45°$)

Starting with joint angles $[\theta_1, \theta_2, \theta_3] = [30°, 120°, 80°]$, and 0 joint velocity for each joint, the task is to catch the ball thrown from $X_A = 3m$, at $V_0 = 10ms^{-1}$ and $\alpha = 45°$. Here several idealised assumptions are made: the ball experiences no air drag and the end-effector can instantaneously catch the ball when it collides with it.

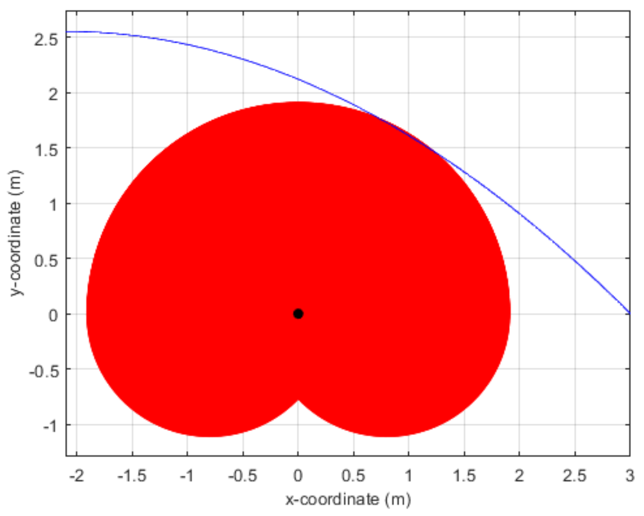
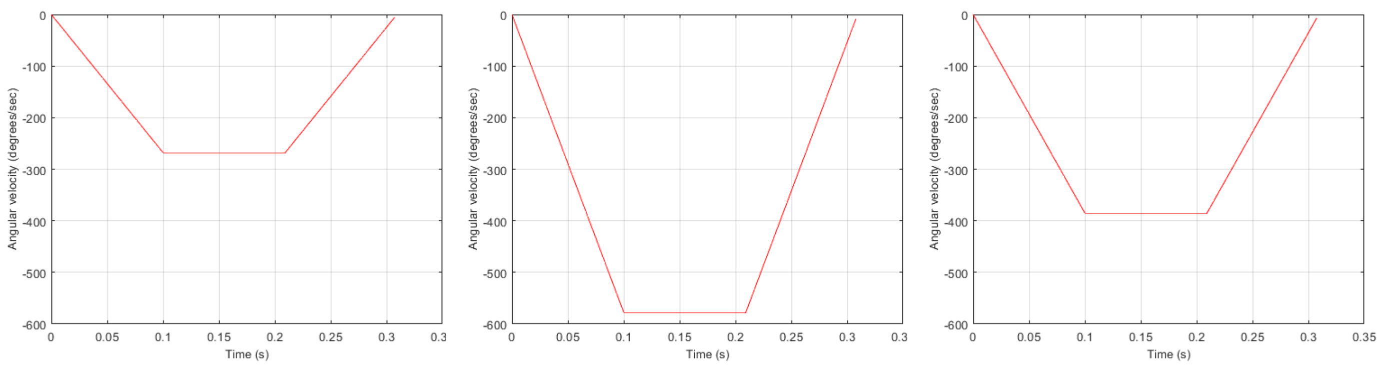


*Figure 6: Trajectory of the ball shown in blue, reachable region in block red, origin in black*

Figure 6 shows there is clearly only a small range of joint angles that can achieve catching the ball (ball trajectory positions in the reachable range). Calculations show that the ball enters the reachable range at time $t_1 = 0.2643s$ and leaves it at $t_2 = 0.3078s$.

With the starting position of the manipulator roughly matching that of figure 2 and the end-effector positions that would lead to catching the ball visible in figure 3, the point that the ball leaves the reachable range is a good place to catch the ball and what was chosen. This point is ($x_3 = 0.823, y_3 = 1.712$). This is because most positions on the ball trajectory will require $\theta_2 \simeq \theta_3 \simeq 0$, and picking the position as far anti-clockwise as possible reduces the angle that $\theta_1$ needs to rotate. Also it gives the manipulator the longest time possible to move to the location required. The joint angles required for this position are $\boldsymbol{\theta} = (-25.7°, 0°, 0°)^T$.



$\theta_1(t)$ $\theta_2(t)$ $\theta_3(t)$

Figure 7: Chosen trajectory joint angular velocity profiles: $\dot{\theta}_i(t)$ for $i \in \{1, 2, 3\}$

$$\dot{\theta}_i(t) = \begin{cases} \dfrac{\dot{\theta}_{i,max} t}{0.1} & : 0 < t < 0.1 \\ \dot{\theta}_{i,max} & : 0.1 < t < (t_2 - 0.1) \\ \dfrac{\dot{\theta}_{i,max}(t_2 - t)}{0.1} & : (t_2 - 0.1) < t < t_2 \\ 0 & : \text{Otherwise} \end{cases}$$

$$(\text{where } \dot{\theta}_{i,max} = \frac{\delta\theta_i}{t_2 - 0.1}, \text{ where } \delta\theta_i \text{ is the change in } \theta_i)$$

Trapezoidal velocity profiles were chosen as they strike a good balance between minimising velocities and accelerations and being practical in the sense that they don't rely on unrealistic assumptions regarding the actuators.

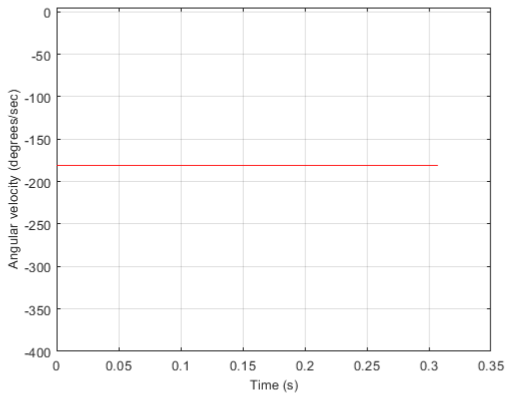
One effect that would be present in reality but is ignored here is the momentum of the ball. If taking this into account, better solutions might have the end-effector moving at a similar velocity to the ball when catching it, to reduce the relative velocity, which in turn reduces the impulse felt by the manipulator. This would require catching the ball at a point on its trajectory inside the reachable range, not on the edge.

## 2.4 Finding Optimal Trajectories

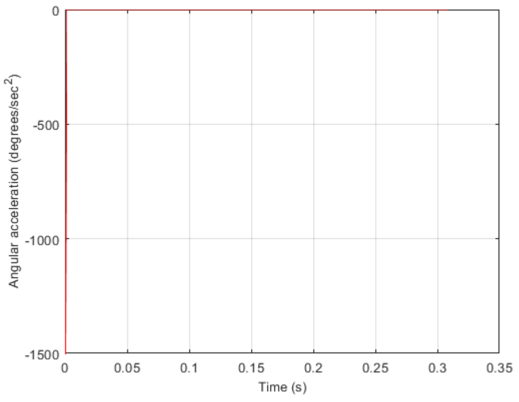
The joint angle changes required for the end effector to move from point $A$ to $B$ in time $\delta t$ can easily be calculated by inverse kinematics. However, there are an infinite number of possible trajectories to make this move in this time. To find an optimal solution, first we must decide what it is we want to optimise with respect to.

**Minimising Joint Angular Velocity**

Minimal joint angular velocities have the optimal trajectory with a constant value angular velocity profile. This means their acceleration profile is a negative delta function at $t = 0$. This can be seen in figure 8.



a)



b)

*Figure 8: Joint 1 angular velocity $\dot{\theta}_1(t)$ & acceleration $\ddot{\theta}_1(t)$ graphs for min velocity*

The plots of $\dot{\theta}_2(t)$ and $\dot{\theta}_3(t)$ would share the same shape, but have different constant values of angular velocity. The joint angular velocity values are:

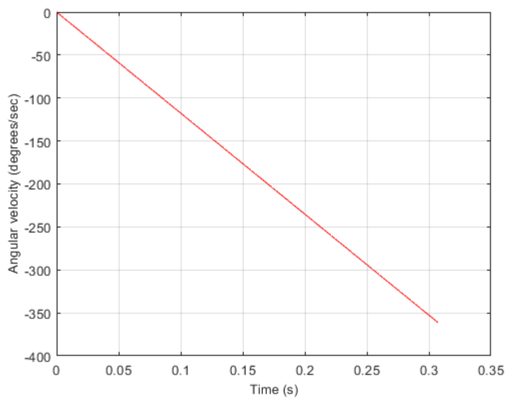
$$\dot{\theta}_1(t) = -181^\circ s^{-1}$$
$$\dot{\theta}_2(t) = -390^\circ s^{-1}$$
$$\dot{\theta}_3(t) = -232^\circ s^{-1}$$

The $\dot{\theta}_1(t)$ and $\dot{\theta}_2(t)$ values occur when reaching the point the ball leaves the reachable range (joint angles $\boldsymbol{\theta} = (-25.7^\circ, 0^\circ, 0^\circ)^T$), while $\dot{\theta}_3(t)$ occurs when the ball is inside the reachable range, with joint angles $\boldsymbol{\theta} = (-31.1^\circ, -3.7^\circ, 13.8^\circ)^T$.

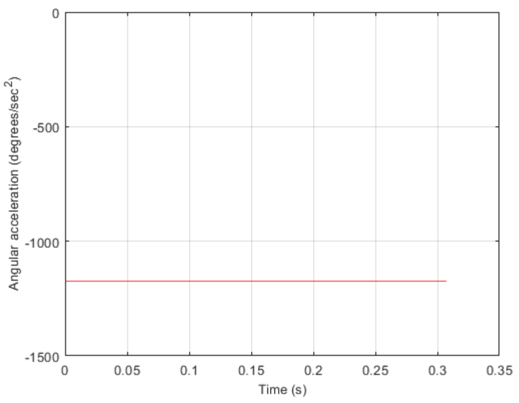
These trajectories aren't practical as the acceleration profiles show they require a (negative) delta function of torque, which cannot be realised with actuators in practice.

## Minimising Joint Angular Acceleration

In a similarly simple way, the minimum angular accelerations required have a flat profile. The velocity and acceleration plots for $\theta_1$ for this scheme are shown in figure 9.



a)



b)

*Figure 9: Joint 1 angular velocity $\dot{\theta}_1(t)$ & acceleration $\ddot{\theta}_1(t)$ graphs for min acceleration*

Again, the best final joint positions for each

$$\ddot{\theta}_1(t) = -1175^\circ s^{-2}$$
$$\ddot{\theta}_2(t) = -2533^\circ s^{-2}$$
$$\ddot{\theta}_3(t) = -1621^\circ s^{-2}$$

## 2.5 Using Dynamic Control

If dynamic control were used for this problem, the algorithm would need to know:
- Mass and moment of inertia of each link and end-effector
- Frictional forces and torques at each joint
- Max torque of each motor
- Latency from command to execution

The difference between dynamic and kinematic control has already briefly been discussed in section 1.2. The crucial difference is dynamic control is more precise, but require more prior information about the environment and more computation.

## 2.6 Designing a Ball-Catching Planning Algorithm

How should a planning algorithm for variable $\alpha$ function? As discussed in section 1.3, in planning problems, it is generally more computationally practical to use discrete methods to solve continuous problems. As catching a ball in 2D space is continuous, this is the design that was chosen.

However, the problem had insufficient constraints and no clearly defined objective function to plan optimally. Thus the design here is based upon these assumptions:

- The ball must be caught above ground level
- The actuators have no torque or angular velocity limits
- Planning algorithm has sizeable, but finite, computation
- Planning algorithm has access to ground truth of ball starting position, velocity and $\alpha$
- Planning algorithm should attempt to minimise the highest average joint velocity

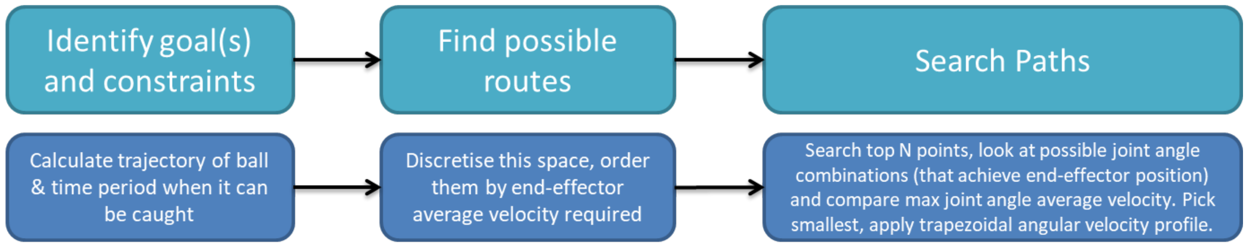
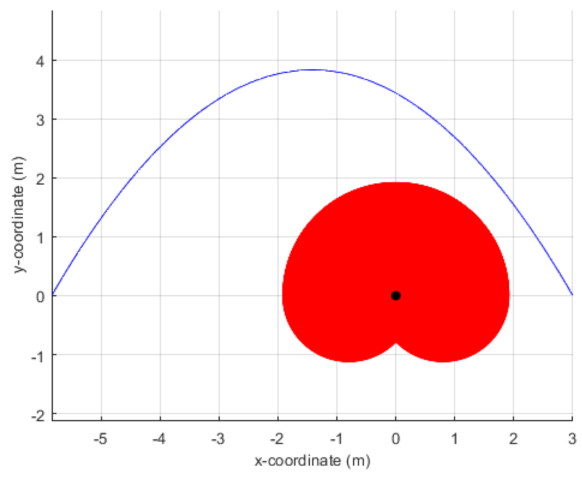
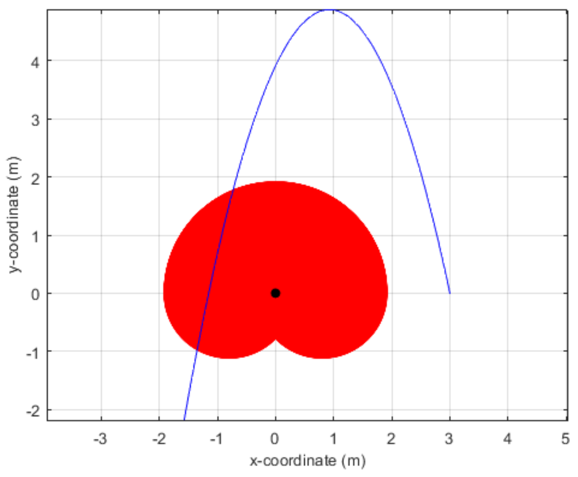

Identify goal(s) and constraints
Find possible routes
Search Paths
Calculate trajectory of ball & time period when it can be caught
Discretise this space, order them by end-effector average velocity required
Search top N points, look at possible joint angle combinations (that achieve end-effector position) and compare max joint angle average velocity. Pick smallest, apply trapezoidal angular velocity profile.

*Figure 10: Flow chart showing discrete planning algorithm design*

The algorithm is split into 3 distinct phases. Calculating the ball trajectory simply looks at the times when the ball enters the reachable range and when it exits, or hits the ground inside it. The next phase is to discretise uniformly by time (perhaps every $0.02s$) to get a finite set of end-effector points to evaluate. Then, by considering the current end-effector position, and these positions and their associated times, work out the average velocity of the end-effector required to catch the ball



$\alpha = 78°$



$\alpha = 78°$

*Figure 11: Reachable range and ball trajectories for variable $\alpha$*

and order the set with the lowest average velocity top.

The last phase does a depth-first search through the top N points on this ordered list, looking at (also discretised) possible joint angle combinations that achieve the end-effector position required. Compare the maximum joint angle average velocity ($l_\infty$-norm of $\boldsymbol{\theta}$ divided by the time) for each set of feasible joint angle combinations and choose the one with the smallest value. Finally the algorithm generates a trapezoidal angular velocity profile for each joint that achieves the angle change in the time required, using $t_\beta$ as the time to accelerate and deccelerate at the start and end. This is another hyperparameter to be set.

The fact that this algorithm only searches the closest (in cartesian space) points means that one cannot ensure that the highest average join velocity is minimised. However, it is a heuristic that should provide reasonable, if not always optimal, results. However it is a heuristic that can easily be calculated and prevents searching the entire set of possible points. This reduces the computational complexity significantly. N can be tuned depending on the amount of computation available.

## 2.7 Catching the ball at variable angle $\alpha$

Applying this algorithm to the values of $\alpha = 60°$ and $\alpha = 78°$, first checks the entry points into the reachable range. For $\alpha = 60°$ the algorithm fails to find any points, as it never enters the reachable range, as can be seen in figure 11. This means the algorithm terminates here and makes no movements.
However, for $\alpha = 78°$, the ball clearly does enter the reachable range, so the algorithm discretises the points at uniform timesteps from the ball entering the reachable range to hitting the ground. The nearest of these points is when $t = 1.9164s$, $x = -0.9844m$, $y = 0.7498m$. This point is achieved optimally (has smallest value of the $l_\infty$-norm of $\boldsymbol{\theta}$) when $\boldsymbol{\theta}= (90°, 54.3°, 106.2°)^T$. Therefore, $\delta\boldsymbol{\theta}= (60°, -65.7°, 26.2°)^T$This manipulator position is shown in figure 12. The hyperparameters were set to $t_\beta = 0.5$ and $N = 1$ for ease of calculation. The joints all follow the trapezoidal velocity profiles, defined:

$$\dot{\theta}_i(t) = \begin{cases} \dfrac{\dot{\theta}_{i,max} t}{0.5} & : 0 < t < 0.5 \\ \dot{\theta}_{i,max} & : 0.5 < t < 1.42 \\ \dfrac{\dot{\theta}_{i,max}(1.92 - t)}{0.5} & : 1.42 < t < 1.92 \\ 0 & : \text{Otherwise} \end{cases}$$

The values of $\dot{\theta}_{max}$ are (THIS IS UNFINISHED)

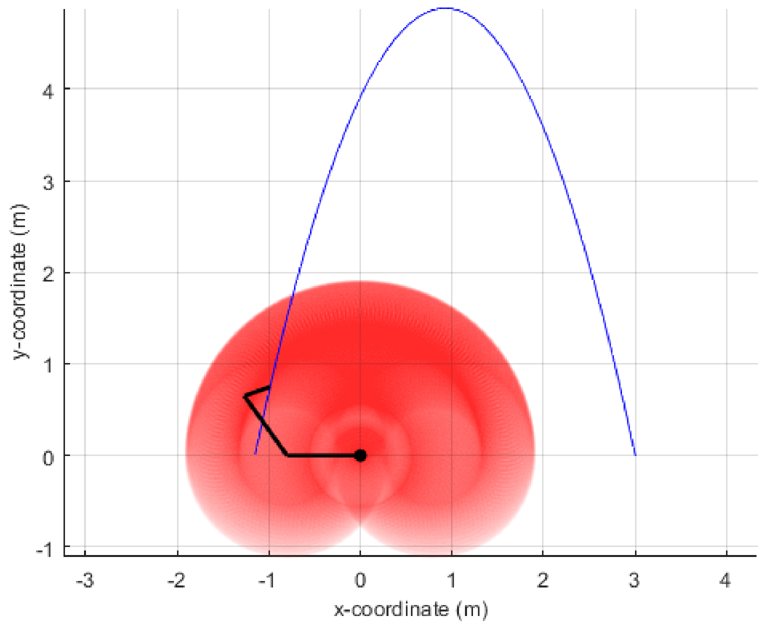


*Figure 12: Final position of manipulator after following planned trajectory*

## References

LaValle, S. M. (2006). *Planning algorithms.* Cambridge university press.