Implementation of a richer library of movements for Babybot using an accurate method to generate trajectories

Project Technical Report: **part One**

In the first part of our project we implemented a simple loop to move the arm. We simplified the underlying architecture in order to employ the least number of intermediate steps. Previously only few possible primitives were accessible by the user, our approach, instead, moves the focus on the user, letting him interact with the arm control board via a simple menu.

We achieved such directness by simplifying the arm thread. The arm thread class contains the main loop that interfaces with the control board, continuously reading the arm joints' positions, velocities and torques. The class contains also the methods that allow making these data visible outside the class itself to be modified as needed. The new arm thread class contains a simplified version of the methods used to initialized and release the arm as well. Such methods allocate and de-allocate the structure used by the YARP arm class. They also calculate the gravity loads used to simulate a human arm. Finally, both methods move the arm into an initial safe position and then park it on the table with a smooth and slow movement (obtained by setting appropriate gains).

The previous reaching protocol used mainly two routines to move the arm: a direct command and a sequence of joint positions generated by the trajectory generator. In this first part of our project we use solely a simplified version of the direct command routine. We thus allow the user to choose which basic movement he wants the arm to perform and to "order" the arm to execute the movement provided that the final position can be safely reached.

At this stage, the menu offers the user the chance to print the current arm position, velocity and torque (in the joint space), to save such information, to move the arm to the last position saved (if it's the current one, no movement is performed and a message is displayed) or to quit the arm loop (ending the arm thread). In the next project step we will use this information to enrich the menu with the option of following various trajectory protocols before letting the robot exploring the library of possible trajectories to a target point while learning which the optimal one is given a certain final task.

The first part of this work has been slowed down by the necessity of learning the previous libraries of methods used to implement reaching and grasping. It was necessary to review and understand most of the code previously generated in order to generate new code, guided and inspired by the previous one.