Incorporating tasks in the dynamics of robotic arms

W.J. Wolfslag∗ and M.C. Plooij∗ and I. Staal and M. Wisse∗

Abstract—We aim to optimize the dynamics of robotic arms, based on the task they have to perform. This will make them faster and cheaper and will lower their energy consumption. We focus on repetitive motions and in particular on rest-to-rest motions, which are used in e.g. pick and place tasks. Incorporating the tasks in the dynamics involves optimizing the mechanical configuration, the mechanical parameters and the control. The combination of these three aspects determines the performance. In this paper we discuss four approaches that we have used to design robotic arms: manual design with optimal feedforward control, parameter optimization with optimal feedforward control, cooptimization of feedback control and mechanical parameters and the use of evolutionary algorithms to design the mechanism from a task description.

I. INTRODUCTION

Much research in robotics aims at making robots perform complex motions, while integrating 3D sensing and vision into the robot control. At the same time in industry, many tasks that are much simpler than those tackled in research are still done by humans, for one simple reason: robots are too expensive, slow and energy inefficient to replace humans in smaller enterprises. The prototypical problem is a basic pick and place (rest-to-rest) motion, for instance in packaging vegetables. Researching such simple rest-to-rest motions can therefore have an immediate impact on industry. Furthermore, mastering these simple motions will allow them to be used as better building blocks in robots performing more complex tasks.

The question remains what topics should be researched. Our philosophy is to incorporate tasks in the dynamics of robotic arms. By matching the dynamics with the task (specifically by using springs to store and release energy at appropriate times in the motion), robotic arms will use less energy, will be faster and cheaper. This incorporation involves three aspects (see Figure 1): the mechanical configuration, the mechanical parameters and the control. Together, these three aspects determine the performance of a system. In this paper, we will discuss the three approaches we have used to address these three aspects and how we envision the design process of the future.

A first approach is manual mechanical design. If we go to the simplest case, a single degree of freedom, the characteristics of the spring can be described qualitatively, and a mechanism can be manually designed to meet that qualitative description. This design process and the resulting mechanisms are discussed in section II, which is based on previously published work [9].

But such a design method has some serious drawbacks. First, there is the question of optimality. The mechanism resulting from a manual design process will have a number of parameters, which are unlikely to be chosen optimally in a manual design process. Furthermore, the mechanism itself cannot be assigned a performance measure such as energy efficiency or speed. Such a measure only arises after the mechanism is combined with a controller. To choose optimal parameters, we therefore used numerical optimization techniques to co-optimize the mechanism parameters and controller. We used two slightly different approaches, which will be discussed in section III, which is based on work previously published [8] and submitted for publication [12].

The second drawback of the manual design approach is that it is difficult to expand into more degrees of freedom. The qualitative description of a spring characteristic becomes harder to define, and matching description to mechanism is also more challenging. Therefore in section IV we will discuss our plan to use evolutionary algorithms to design these mechanisms from scratch. This part will mainly focus on the method to represent mechanisms in a form that evolutionary algorithms can cope with.

Both the co-optimization approach and the evolutionary design approach are phrased as an optimization problem, and therefore depend on some performance measure to optimize. The standard performance measures such as cost, safety, energy efficiency, speed and maximal torque or jerk are clearly important for rest-to-rest motions. However, in an optimization scheme, they are not ideal because they are hard to compute. For safety, and to a lesser extend cost this is clear. The other terms are computable with an accurate mechanical model.

Unfortunately, a validated model is only available after the design process. Estimating friction and effects due to non-rigidity or play is difficult, if not infeasible. Therefore, in the future, optimization schemes should explicitly account for
robustness against variations in these effects. In section III we will touch upon a peculiar result, which is in some part caused by this robustness not being taken into account. In section IV we will briefly describe how we aim to prevent such robustness issues in the evolutionary approach.

In all three the result sections, the robot has only a single degree of freedom. Although such a robot is unlikely to be of much practical use, it is an important benchmark. Because it is simple, it is possible to fully understand what is going on, and honestly compare different designs. However, adding additional degrees of freedom is typically not trivial. Therefore, in the original papers discussing these results, we have included analysis of two-DOF robot arms. While still simple, this arm is dynamically almost equivalent to the SCARA type arms which are used in industry. Furthermore, these two-DOF arms provide a good balance between complexity and simplicity, which allows to really test the design on a challenging problem, while being able to interpret the results.

II. MANUAL DESIGN

Often, a pick and place task in industry will be highly repetitive. In effect, the robot is asked to repeatedly move from a standstill position near one end of its workspace, to a position on the other end, and back. Such a motion is reminiscent of the oscillation of a mass-spring system. It is intuitive that using such a spring to power the motion could lead to energy savings. However, there is one key difference between an oscillating mass-spring system and the desired movement of the robot: the robot has to stand still for a while at its extreme positions.

When using a linear spring, some form of actuation is then needed to counteract the effect of the spring at these endpoints, in order to hold the robot at its place. So ideally, we would like something that behaves like a linear spring in between the end-positions, but delivers no force at the end-positions themselves. A spring with the potential energy sketched in Figure 2a, does this. Around the 0-position (halfway the end-positions), the potential energy looks like a parabola, so like a normal linear spring. Near the end-positions, the slope of the potential energy is 0. As the torque acting upon the robot is this slope, the force is 0. So the given potential energy can power the motion, while allowing the robot to stand still at its endpoints.

But, a mechanism is still required that produces this potential energy. To design such a mechanism, we used a linear extension spring in combination with a mechanism that provides the kinematics to fit the desired potential energy, e.g., a four-bar-linkage. We made a systematic overview of 9 relatively simple kinematic mechanisms and excluded 7 mechanisms that did not fit the desired kinematics. From the final two concepts, we chose the one which is shown in Figure 2b, because it was easiest to construct.

A prototype was made and using optimal feedforward control a comparison was made between a robot with and without this spring mechanism. We concluded that the mechanism lowered the energy consumption by 20%. See [9] for further details.

III. OPTIMIZING CONTROLLER AND MECHANISM PARAMETERS

With the spring mechanism in Figure 2, a significant improvement in energy efficiency was made. However, the question remains, how much improvement is possible? Translated to this particular problem, we see that the mechanism has a number of parameters, e.g., spring stiffness and zero-length, radii of the two pulleys and distance between the pulleys. The most improvement is found when these parameters are chosen optimally. In this section we will discuss two approaches which find the optimal parameters. In both approaches, the key problem is to find a performance measure for the mechanism. This is primarily a problem, because the mechanisms performance cannot be computed before a controller is known. Furthermore, for fair comparison, such a controller should be optimal itself as well. Therefore we have to co-optimize mechanism and controller. To solve this, we used two approaches, which we will briefly discuss below. A more detailed explanation of these approaches can be found in [11] and [8] respectively.

The first approach we took was to use optimal feedforward control [6] to find the optimal controller, given a set of mechanism parameters. To show that the mechanism does not just improve the energy efficiency, we aimed to minimize the duration of the motion, given torque limits on the motor.

A potential problem of this approach is that this will optimize the mechanism for one very specific motion. As a result, the performance of the robot might become very sensitive to the specific motion parameters. To avoid this, and make a more generally useful mechanism, the performance measure was extended by taking multiple goal positions. So for every goal position in a target range, the duration of the quickest motion from a fixed start-position to that goal position was computed. The total performance measure was...
the integral of these durations over the range. Figure 3, shows the resulting movement times over the range of goal positions. We clearly see that there is a large improvement over the original mechanism, particularly when we allow a larger potential energy at the end-positions.

The second approach is more focussed on the idea of co-optimization. Inspired by the idea of symbiosis in biological evolution, we investigated whether it was possible to use a similar type of evolution to co-optimize controller and mechanism.

We used two evolutionary algorithms: CoSyNE[1] and CMA-ES[7]. CoSyNE does not use one representation for both controller and mechanism, but to use two separate representation, which evolve parallelly in an environment where each controller is linked to a mechanism to determine the fitness function of the combination. CMA-ES uses the more standard approach of using a single representation. For implementation of these algorithms, a parametrized controller is required.

A fuzzy-logic was chosen, due to easy interpretation of the resulting controller. And again to show the versatility of the basic mechanism design, we optimized for the maximal torque required during motion, which includes a brief moment of standing still at the end-point. Also again, to avoid robustness issues with optimizing for a motion that is too specific, a number of goal states were picked. The final performance measure was now the maximum minimal torque over the motions towards these goal states.

Figure 4 shows the resulting potential energy for various runs of the two algorithms. It is clear that CMA-ES converges to two different local optima. The results from the CoSyNE algorithm on the other hand have a much larger spread. In [8], this is attributed to CMA-ES changing the parameters more frequently and gradually, as opposed to the more infrequent, jumping changes in the parameters that CoSyNE uses.

An interesting feature of the optimized potential energy curves for both evolutionary algorithms is that the potential energy curve is nowhere flat. This is largely explained by the performance measure: a torque at standstill is not punished, as long as this torque is smaller than the maximal torque needed during the motion itself. A problem occurs because the exact optimum of this slope depends on the friction in the model, as explained below. We used a model with Coulomb friction, so the friction helps the controller to keep the robot at a stand-still. Optimally, the sum of this friction and the stand-still torque should equal the maximal torque during the motion. The problem here is that the Coulomb friction cannot be accurately estimated before the mechanism is built, and is not even trivially determined after the building. Finding a way to make the optimization robust against these uncertain parameters is a key part in our future work.

IV. EVOLVING MECHANISMS FROM SCRATCH

The previous two optimization based design strategies relied on one assumption: the abstract level concept of two revolute joints connected by a pulley and a spring is the optimal concept for a spring mechanism for pick and place tasks. Although this particular mechanism might be near-optimal, it is difficult to judge whether this is the case. If we go to more challenging tasks, for instance an arm with two degrees of freedom, it is already challenging to come up with any concept, let alone an optimal one. To counter this, we need to incorporate this abstract concept design into our optimization.

Therefore we have begun to use evolutionary algorithms to evolve complete concepts, and not only the parameters of a human-designed concept. The remaining challenge here is to design a representation and a repair function that allows the evolutionary algorithm to work efficiently. Since this work is not finished, we only discuss our work on the properties we would like to see in our representation of the mechanism.

To design a complete concept, the idea is to combine basic building blocks, such as rigid bodies, hinges, springs and dampers. Because we do not know the number of building blocks, their parameters values (e.g. lengths and spring stiffness) or their connections, the representation needs to be highly flexible. This is also the reason an evolutionary algorithm is used for this optimization problem: it is one of the few methods that can cope with a representation that is changing in size and structure.

Our representation is inspired by Darwin2K [4], which represents robot manipulators in an object-oriented style. An
alternative representation that also meets the flexibility demand [3], is less suitable for our problem in which the building blocks are dissimilar and not all connections are allowed.

One problem with Darwin2K is that it views the connections between the various building blocks as a graph, and the representation assumes this graph is acyclic. This prevents parallel structures in the mechanisms, which for instance disallows the manually designed concept discussed before, but also any four bar linkage.

To solve this problem, we use a graph structure inspired by [10]. This breaks up the building blocks in two classes: rigid bodies, and connections, such as springs, hinges or gear-ratios. Because all these connections are between two rigid bodies, we can see the mechanism as a graph with the bodies as vertices and the connections as edges. We use the incidence matrix of this graph in our representation. An example mechanism and its representations are shown in Figure 5.

The second demand for the representation is evolvability. By evolvability we mean that standard evolutionary operators such as cross-over and mutation should make sense. This happens if similar genotypes lead to similar phenotypes. The result is that a small mutation leads to a small change in mechanism, which helps evolution [2].

The proposed representation meets this requirement for almost all mechanism parts. Parameters inside the rigid bodies or connections are easily mutated by taking small deviations in their values, or by interchanging them between bodies or connections. The connection graph is also easily perturbed by changes in the incidence matrix, a process particularly suited for a cross-over like operator. This is an advantage over the representation in [5], which is less suited for cross-over. The only concern are constraints, which are typically either on or off, and are therefore difficult to change gradually. Preliminary results indicate that this effect is adequately dealt with by the evolutionary algorithm, but for more complex cases adjustments to the representation might be required.

V. DISCUSSION AND CONCLUSION

In this paper we discussed the importance of rest-to-rest motions of robot arms with one or two degrees of freedom. This importance is due to three reasons: direct impact on industry, potential use as building block for more complex robots and usability as test case or benchmark for optimal task based design.

We also presented our shift from manual design to computer optimized design, showing a spectrum of design approaches. Each approach has different ways to design the mechanical configuration, choose the mechanical parameters and design a controller. These three aspects of a robot are equally important for achieving an optimal design. However, in current design processes, they are typically regarded separately. Therefore, future research should be focused on combining those three aspects in one optimization.

REFERENCES