Dissipatively actuated manipulation

W.J. Wolfslag\textsuperscript{1}, M.C. Plooij, W. Caarls, S. van Weperen and G.A.D. Lopes

Abstract

This paper addresses the design of control systems whose actuation can only dissipate energy. Such systems provide intrinsic safety, and can be used in scenarios where energy is supplied by external entities and point-stabilization is possible with only energy dissipation. Three control synthesis methods are proposed that range from model-based to a learning approach and their validity is demonstrated on a passively controlled manipulator performing a positioning task. These three methods are the Zero Control Velocity Field, Monte-Carlo Tree Search and Reinforcement Learning. The simulation results are corroborated by experiments on a physical two link manipulator.

Keywords: robotic manipulation; dissipative actuation; non-linear control; receding horizon control; reinforcement learning

1. Introduction

On many robotic arms, actuators actively provide the power needed to achieve motion. Designers often aim for the cheapest or lightest actuator that suffices the power requirements, in order to reduce the cost and weight of the arm. This paper takes this aim to an extremum and proposes control systems for a manipulator that does not use active actuators to power the motion. In such a manipulator, motions can only be influenced by clutches and dissipative components such as brakes. As a case study for this idea, the system shown in Figure 1a is examined: a two DOF manipulator in the vertical plane which picks up objects and places them at a lower height. While doing so, the controller can dissipate energy but cannot add energy to the manipulator. Such a system shows resemblance with a skier (see Figure 1b), who can steer and brake by dissipating energy while going down to end up at a desired location.

Systems with solely dissipative components have several advantages. First, they exhibit intrinsic safety: such systems cannot make unexpected motions caused by unexpected inputs. Therefore, the motions are very transparent for the user, causing the system to be safer. The advantage of intrinsic safety is that it does not rely on active control, as in [1], or clever trajectories enforcing safety during control failures, as in [2], both of which can potentially fail in operation. Obtaining intrinsic safety was the prime motivation for previous research on dissipatively actuated systems in the field of haptic devices [3, 4, 5], particularly for rehabilitation purposes[6, 7]. Second, manipulators without motors are cheaper in both purchase (actuators are expensive) and usage (lower energy costs). The energy considerations prompted research in walking robots to consider purely unactuated robots [8], and derived robots which use very limited actuation combined with the dynamic properties of walking [9, 10, 11]. Finally, dissipatively actuated manipulators could lead to lightweight designs.

The goal of this paper is to introduce the concept of dissipatively actuated manipulation. This concept turns out to be challenging from a controls perspective, as the resulting systems do not fulfill the required assumptions of the traditional control ap-
proaches for nonlinear input-affine models. Therefore we compare three controllers that do not make these assumptions: a Zero Control Velocity Field (ZCVF) controller, a Monte-Carlo Tree Search (MCTS) controller and a Reinforcement Learning (RL) controller. These three controllers are chosen as they form a scale from a mostly model-based towards a purely numerical approach. As such, they represent three main paradigms in control: model-based, receding horizon and learning.

Past research into the control of dissipatively actuated systems has largely been guided by the requirements of haptic devices. Two prime topics of interest are the regulation of admittance using dissipative actuators [12, 13], and the creation of position dependent steering forces at an end-effector [6, 14]. In both these cases the desired behaviour is expressed locally, which is not possible in our tasks: position control.

More global behaviour was obtained in research on walking-guidance by means of wheeled robots [15]. However, as the desired behavior of that robot was guidance, the control system was based on a user supplying a force in approximately the right direction. Furthermore, the main external force, gravity, was compensated for using active actuators. For these two reasons their approach is not directly applicable to the problem here. The zero-control-velocity-field controller (discussed in section 4), is the most similar to their approach, as it tries to steer towards a preplanned trajectory.

Outside of haptic guidance, dissipative actuators are less commonly used. In underactuated robot manipulators, brakes have been used as a locking mechanism, allowing the robot to reach a desired configuration by sequentially manipulating and locking the unactuated degrees of freedom [16, 17, 18]. Although such a decoupling mode could be used in a fully dissipatively actuated system, the decoupling structure severely limits the possibility of control.

The rest of the paper is structured as follows. Section 2 formulates the dissipatively actuated manipulation problem mathematically. Section 3 describes the test case chosen to test the controllers: a two DOF manipulator in the vertical plane. Sections 4, 5 and 6 discuss the three controllers compared in this paper: ZCVF control, MCTS control and RL control. The paper ends with a discussion in Section 7 including a comparison between the three controllers and a conclusion in Section 8.

2. Dissipatively actuated systems

Consider a class of mechanical systems in which the controller cannot add energy to the system. For mechanical systems, the energy is expressed as [19]:

$$H(q, p) = \frac{1}{2} p^T M(q)^{-1} p + V(q)$$  \hspace{1cm} (1)

where $q \in \mathbb{R}^n$ the generalized coordinates, $p = M(q)\dot{q}$ the generalized momenta, $M(q)$ the positive definite mass matrix, and $V(q)$ the potential energy.

The equations of motion can now be expressed in port-

Hamiltonian form [20]:

$$\begin{align*}
\begin{pmatrix}
\dot{q} \\
\dot{p}
\end{pmatrix} &=
\begin{pmatrix}
0 & I \\
-I & -R
\end{pmatrix}
\begin{pmatrix}
\nabla_q H \\
\nabla_p H
\end{pmatrix} +
\begin{pmatrix}
0 \\
I
\end{pmatrix} u \\
\xi &= (0, I) \begin{pmatrix}
\nabla_q H \\
\nabla_p H
\end{pmatrix}
\end{align*}$$  \hspace{1cm} (2)

where $u \in \mathbb{R}^n$ are the input torques, $R$ a positive definite matrix representing mechanical damping (e.g. friction), and $\xi = \nabla_p H = \dot{q}$, is the output of the system.

From this formulation, one can show that the system is passive with the total energy $H$ as storage function, as can be seen by expressing the time derivative of $H$:

$$\dot{H} = -\dot{q}^T R \dot{q} + \xi^T u \leq \xi^T u$$  \hspace{1cm} (3)

The input does not add energy to the system as long as $\xi^T u \leq 0$, or equivalently:

$$\dot{H} \leq -\dot{q}^T R \dot{q}$$  \hspace{1cm} (4)

In this case, the controlled system is passive with respect to the original Hamiltonian, even if there is no damping ($R = 0$). Denote systems (respectively controllers) that satisfy eq. (4) as dissipatively actuated systems (respectively dissipatively actuated controllers). Such systems should be distinguished from dissipative control systems, described in for instance [21],[22], in which both the controller and the uncontrolled system meet a demand similar to Eq. 3. Do note that a dissipatively actuated system is always a dissipative control system, the converse is not true.

A more strict demand is

$$\xi_i^T u_i \leq 0 \quad \forall i = 1, \ldots, n$$  \hspace{1cm} (5)

which gives rise to elementwise dissipatively actuated systems. This distinction is important in implementation, because elementwise dissipatively actuated systems can be constructed using only brakes, whereas dissipatively actuated systems in general require clutches. The implementations discussed later in this paper are all elementwise dissipatively actuated controllers.

The control goal is dissipatively actuated manipulation: steering a dissipatively actuated system to a desired state. The first challenging aspect of dissipatively actuated manipulation can be understood from eq. (4). Since the energy in the system must always be decreasing, the problem of point stabilization becomes challenging: if the energy of the desired final state is higher then the total energy of the initial condition, then there exists no solution.

Even neglecting such reachability issues, the design of dissipatively actuated controllers is not solved by standard methods. To see this, consider the following structure of an elementwise dissipatively actuated controller:

$$u = -D(q, \dot{q})\dot{q}$$  \hspace{1cm} (6)

where $D$ is positive diagonal. If instead of this diagonal constraint, $D + D^T$ is constrained to be positive definite, a normal dissipatively actuated system is obtained, as Eq.4 is then satisfied. In either case, if $D$ is allowed to be discontinuous, all
Dissipatively actuated manipulation

Table 1: The model parameters of the two DOF arm.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Symbol</th>
<th>Value</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coulomb friction</td>
<td>μc1, μc2</td>
<td>0.25, 0.08</td>
<td>Nm</td>
</tr>
<tr>
<td>Viscous friction</td>
<td>μv1, μv2</td>
<td>-0.024, -0.009</td>
<td>Nms/rad</td>
</tr>
<tr>
<td>TD-friction</td>
<td>μt1, μt2</td>
<td>32, 22</td>
<td>%</td>
</tr>
<tr>
<td>Inertia</td>
<td>J1, J2</td>
<td>0.0364, 0.0808</td>
<td>kgm²</td>
</tr>
<tr>
<td>Mass</td>
<td>m1, m2</td>
<td>0.8, 1.0</td>
<td>kg</td>
</tr>
<tr>
<td>Length</td>
<td>l1, l2</td>
<td>0.38, 0.55</td>
<td>m</td>
</tr>
<tr>
<td>Position of COM</td>
<td>l₁g1, l₂g2</td>
<td>0.07, 0.25</td>
<td>m</td>
</tr>
<tr>
<td>Motor constant</td>
<td>kt1, kt2</td>
<td>25.9, 25.9</td>
<td>mNm/A</td>
</tr>
<tr>
<td>Gearbox ratio</td>
<td>n1, n2</td>
<td>1:22.5, 1:82.5</td>
<td>rad/rad</td>
</tr>
</tbody>
</table>

(elementwise) dissipatively actuated state feedback controllers can be written in this form.

The constraints on $D$ have two important consequences for model based controller design: First, they mean the system is no longer small-time locally controllable [19], as can be seen from the reachability argument above. Secondly, while the system is input affine, its control space is not proper, i.e. 0 is on the boundary of the control space. Unfortunately, most results on input-affine systems (see [19][21]) require the input space to be proper, or even to be $\mathbb{R}^n$. As such, the requirements for most nonlinear model-based control synthesis techniques (e.g. feedback linearization, backstepping or sliding mode control; see [21]) are not met. Furthermore, when using a flatness based approach [23], the constrained space is not readily expressed in bounds on the flat output and its derivatives. This makes a flatness based approach unappealing as well.

It is concluded that the problem of dissipatively actuated control is not readily solved by standard model-based control design methods. In the remainder of the paper three approaches are discussed that have a significant degree of numerical planning contained in them (this is also the case for other types of more challenging plants such as nonholonomic systems).

3. Case study

To test the three proposed approaches, a pick and place task for a two DOF arm in the vertical plane is used. Before discussing these approaches, the hardware setup is described, how it is modeled and the task it has to perform.

3.1. Robotic manipulator

Figure 2 shows a picture of the two DOF robotic arm [24]. The DOFs are created by two 18x1.5mm stainless steel tubes, connected with two revolute joints. A mass of 1 kg is connected to the end of the second tube, which represents the weight of a gripper plus a product. The motors are placed on a housing and AT3-gen III 16mm timing belts are used to transfer torques within the housing. The joints are actuated by Maxon 60W RE30 motors with gearbox ratios 1:18 and 1:66 respectively. The timing belts provide an additional transfer ratio of 5:4 on both joints. The parameters of this robotic arm are listed in Table 1.

The friction in the joints is modeled as a combination of viscous, Coulomb and torque dependent friction (TD friction in Table 1):

\[
T_f = -\mu_v \cdot \omega - (\mu_c + \mu_t \cdot |u|) \cdot \text{sign}(\omega) \quad \text{for} \quad \omega \neq 0 
\]

\[
T_f = - (\mu_c + \mu_t \cdot |u|) \cdot \text{sign}(u) \quad \text{for} \quad \omega = 0
\]

The standard mechanical equations are derived from the port-Hamiltonian model presented in (2), using the Legendre transformation, to obtain:

\[
M(q)\dddot{q} + C(q, \dot{q})\dot{q} + \nabla_q V = u + T_f
\]

where $M$ is the positive definite mass matrix, $C$ the matrix representing the Coriolis and centrifugal forces, and $V$ the potential energy due to gravity. Further details of these equations are omitted from this paper due to length. Note that because of the parallel mechanism created by the timing belt to the second joint, the angle of the second arm is taken as the absolute angle, i.e., relative to the world frame.

3.2. Pick and place task

The manipulator has to perform a pick and place task with one pick position and four place positions (see Figure 2). Every
motion starts at the pick position while knowing the place position it has to go to. When the place position is reached, the arm is transferred back to the pick position. In the sketch of the idea in Figure 1, this transfer is done by a spring once the package is dropped and the total mass of the manipulator reduces. The actual robot setup used in this paper does not include such a spring and thus the arm is moved up using the motors.

For mechanical systems, disturbance handling is often an important issue. In the design of the three controllers, these issues are not explicitly addressed, as the focus is on the challenge of controlling a robot arm to a desired position using only dissipative actuators. The capability of handling large disturbances is discussed in Section 7 as part of the reachability question, as it relates directly to the range of motions that the robot can perform.

4. Zero Control Velocity Field

4.1. Steer towards minimal control

The first approach proposed is based on a simple heuristic: when control authority is limited, steer towards a trajectory that requires little or no control, and then keep following that trajectory using the natural dynamics. The idea behind this heuristic is that if a disturbance would act upon the system in a trajectory that uses no (or very limited) control, it is likely that enough control authority remains to steer back to the desired trajectory. Contrast this to the case where all control authority is required to follow the trajectory, which is likely to make it impossible to recover from a disturbance.

The above heuristic is applied using a three staged approach. The first stage is to find trajectories towards the goal position that use a control action that does not come close to violating the constraint in eq. (5). The second stage is a velocity field for which the previously found trajectories are attractive. The third stage is the design of a controller that tracks the velocity field. This final controller is then used during the motion. In the description below it is shown how these three stages are incorporated into the controller for the vertical pick and place task. The complete control procedure is summarized in Algorithm 1.

4.2. Implementation

Stage 1, trajectory

To find the minimum control trajectories backward simulation is used. A trajectory \( q_{\text{traj}}(t) \) is computed by starting at the goal position and simulating backwards in time, using a feedforward signal (i.e. a constant gain damping coefficient). This trajectory is represented by letting the workspace \( x \)-position be a function of the workspace \( y \)-position, resulting in the trajectory always going downwards at all time. This also makes the trajectory not cross itself.

In the backward simulation, there are two parameters that have to be tuned. The first tunable parameter is the velocity at the goal position. In a practical application, this goal velocity is typically 0 rad/s. However, a large braking torque towards the end of the motion can easily be used to achieve that velocity.

Therefore any goal velocity can be used to tune the motion in the backwards simulation\(^2\). A final braking step is then used to obtain the actual goal velocity on the hardware. This braking step will clearly dissipate energy, so a standard controller can be used without requiring an actuator that provides energy. Because such a standard controller can be used, this braking step is ignored in the rest of this paper. The second parameter to tune for the backward simulation is the virtual damping coefficient. A diagonal matrix is used to specify the simulated damping, where the damping coefficients are the same for each link, leaving just one tunable parameter. The velocity and damping are manually tuned to obtain trajectories with a low endpoint velocity that do not require large braking torques. Table 2 shows the damping and velocities used for the goal positions in the experiment.

Stage 2, velocity field

A velocity field \( q_{\text{des}} = N(q) \) is created such that \( N(q_{\text{traj}}(t)) = q_{\text{traj}} \), implying that \( q_{\text{traj}} \) is an invariant set for the velocity field. Furthermore, this invariant set is made to be attractive, that is: one must construct a Lyapunov function \( \mu(q) \), which has its minimum on all points of the trajectory, such that \( \mu = \nabla_y \mu(q) N(q) \leq 0 \). The vector field \( N(q) \) is designed by constructing a velocity field in the end-effector coordinates \((x,y)\), and transforming that velocity field back to configuration space coordinates, see Figure 3. First the uncontrolled trajectory is parameterized. This is achieved by using \( q_{\text{traj}} \) to parametrize the \( x \)-position of the endpoint trajectory as a polynomial of its \( y \)-position: \( x(y) = p_x(y) \), and then have an error function \( G(x,y) = x - p_x(y) \). The velocity magnitude is also needed, which is also taken as a polynomial in \( y \): \( \sqrt{x^2 + y^2} = p_v(y) \). Fitting of both these polynomials is done using least squares approximation.

The next step is to define two vector fields \( \vec{f}_{d1} \) and \( \vec{f}_{d2} \). The first, \( \vec{f}_{d1} \), points towards the desired trajectory \( q_{\text{traj}} \). The second, \( \vec{f}_{d2} \), enforces the correct velocity once the system is on the trajectory \( q_{\text{traj}} \). A simple way to define these vector fields is as follows:

\[
\vec{f}_{d1} = -p_v(y) \text{normalize}(\nabla_1 G(x(y),y))
\]

\[
\vec{f}_{d2} = p_v(y) \text{normalize}(\nabla_y (p_x(y)))
\]

\(^2\)If the final velocity given by the control specification is nonzero, a similar reasoning applies. The only change is that the final velocity for the simulation must be larger than the required velocity.
The complete vector field \( f_a(x,y) \) is a weighted combination of the two fields mentioned above. When \( x \) is close to \( p_x(y) \), the vector field steers mostly along the trajectory. Oppositely, when \( x \) is far from \( p_x(y) \), steer happens towards the trajectory. This intuitive approach is implemented using the weighting function \( n(x) = (1 + G(x)^2)^{-\alpha} \), with \( \alpha \in \mathbb{R}^+ \) a parameter that manipulates the trade-off between how quickly the field steers towards the desired trajectory, and how much acceleration is need to follow the vector field. For the experiments presented in this paper \( \alpha = 100 \). The resulting vector field is:

\[
    f_a(x,y) = (1 - n(x))f_{d1} + n(x)f_{d2} \tag{12}
\]

This field is defined in workspace coordinates, and is converted to configuration space by means of the Jacobian of the workspace-configuration space transformation to obtain \( N(q) \).

**Stage 3, field controller**

To control towards the velocity field, an inverse dynamics approach is used [25]. Let \( \dot{N}(q,\dot{q}) = \dot{q} - N(q) \), be the velocity error. Then the input \( u \) is determined by:

\[
    u = -K\dot{N}(q,\dot{q}) - \nabla_q\mu(q) + M(q)\nabla_qN(q)\dot{q} + C(q,\dot{q})N(q) + B(q,\dot{q}) \tag{13}
\]

where, \( K \in \mathbb{R}^{n\times n} \) is a linear feedback-gain, and can be set to any positive definite matrix. For the experiments only the diagonal terms are used, which were tuned manually to obtain:

\[
    K = \text{diag}(30,30). \quad \mu(q) = G(x,y)^2 \text{ is a valid Lyapunov function, satisfying } \nabla_q\mu(q)N(q) \leq 0. \quad B(q,\dot{q}) = \nabla_qV(q) - T(q,\dot{q}), \text{ is a term compensating for gravity and friction.}
\]

Using the candidate Lyapunov function

\[
    L = \frac{1}{2} \dot{N}(q,\dot{q})^T M(q)\dot{N}(q,\dot{q}) + \mu(q) \tag{14}
\]

one can show [25] that the input \( u \) in equation (13) renders system (9) stable at \( \dot{N}(q,\dot{q}) = 0 \). Note that the control law described above is not necessarily a dissipatively actuated controller. Therefore \( u \) is projected on the allowed control space, using unweighed least squares. Finally, because of hardware limitations a torque saturation of 3.5 Nm is used on the first motor and 2.5 Nm on the second motor.

**4.3. Results**

The velocity field for moving towards \( x = -0.3 \text{ m} \) is shown in Figure 3. In this figure, the blue line shows the backwards integrated motion, the red line shows the polynomial approximation of that motion, and the black arrows show the velocity references for the specific positions. Four such velocity fields were obtained, one for each goal position.

Figure 4.a-bottom shows the four resulting motions of the arm in simulation. The figure shows that the arm moves toward the four goal positions, but there is an error in the position at which the arm reaches the ground. The average position error for the four positions is 1.9 (±1.6) % of the length of the arm. Because the arm is nearly 1 m long, this percentage is very close to the absolute error measured in cm.
Dissipatively actuated manipulation

Simulations

Experiments

Figure 4: Simulation and experimental results using the a) zero control velocity fields, b) Monte-Carlo search tree, and c) reinforcement learning. The black lines represent the trajectories towards the four goal positions, which are indicated by black crosses. The blue lines show different positions of the two DOF arm, while moving towards the most left goal position.

5. Monte-Carlo Tree Search

The second proposed approach uses receding horizon control to steer the system in the right direction. This means that at every time step the model is used to plan from the current state to the goal, but only the first action in the plan is taken. This should reduce the requirements on the accuracy of the model because the error is continually corrected on-line.

5.1. Controller

A standard Monte-Carlo Tree Search (MCTS, [26]) method is used, summarized in Algorithm 2. The tree consists of nodes \( \nu \) associated with a state \( s(\nu) \), which is defined as \( s = [q, \dot{q}]^T \). These nodes each have a number of children equal to the number of actions possible in that state. First, a root node \( \nu_0 \) is created at the current state \( s_0 \). Then, while there is still time, a node \( \nu_t \) is iteratively selected from the tree, perform a rollout from that state \( s(\nu_t) \), and back up the resulting cost \( \Delta \) through

Algorithm 2 Monte-Carlo Tree Search

1: function MCTS(s_0)
2: create root node \( \nu_0 \) with state \( s_0 \)
3: while within computational budget do
4: \( \nu_t \leftarrow \text{SELECT}(\nu_0) \)
5: \( \Delta \leftarrow \text{ROLLOUT}(s(\nu_t)) \)
6: \( \text{BACKUP}(\nu_t, \Delta) \)
7: end while
8: return \( a(\text{BESTCHILD}(\nu_0)) \)
9: end function
the tree. When time is up, the action with the lowest cost from the root node is returned.

Node selection is done by traversing the tree using an $\epsilon$-greedy policy, selecting a random child with probability $\epsilon$ and the child that has the lowest cost otherwise. When a node is reached for which not all actions have been evaluated yet, a random unevaluated action $u$ is expanded and the resulting child node $v_t$ is returned. Rollouts are performed by taking random actions starting at $s(v_t)$ until the arm hits the ground or until the evaluation horizon is reached. The total cost $\Delta$ of this rollout is assigned to $v_t$. Then, for all parent nodes $v$ up until the root node, the cost $c(v)$ is set to

$$
\frac{k(v)c(v) + \Delta}{k(v) + 1},
$$

where $k(v)$ is the number of times $v$ has been selected before.

Dissipativity of the control actions is ensured by parametrizing the actions such that they always meet the dissipativity constraint. This can always be done by projecting the action on the feasible actions. For the elementwise case a more elegant approach is to only use actions with a positive value, and multiply the action-value with the sign of the velocity. This can always be done by projecting the action on the feasible actions. The optimal value function is the unique solution to the Bellman equation

$$
Q^*(s, u) = \min_{u'} Q(s', u') + \alpha \delta
$$

where $s'$ is the next state when taking action $u$. $Q$ is estimated by sampling state transitions $(s, u) \to (s', s')$ along a trajectory and updating $Q$ towards minimizing the temporal difference error $\delta$. We use the SARSA algorithm, summarized in Algorithm 3.

### Algorithm 3: On-policy TD control (SARSA)

1: function SARSA
2: obtain initial value function $Q$
3: while not converged do
4: move robot to start position
5: obtain state $s$ and set $u \leftarrow \pi(s; Q)$
6: while floor not reached do
7: apply braking torque $u$.
8: obtain new state $s'$ and set $u' \leftarrow \pi(s'; Q)$
9: $\delta \leftarrow c(s') + Q(s', u') - Q(s, u)$
10: $Q(s, u) \leftarrow Q(s, u) + \alpha \delta$
11: $s \leftarrow s', u \leftarrow u'$
12: end while
13: end while
14: return $Q$
15: end function

In this algorithm, $\alpha$ is a learning rate that sets the parameter of an exponential moving average filter determined by the stochasticity of the problem. To speed up convergence the SARSA($\lambda$) algorithm is used, which not only updates the current state but also a number of previous states based on their eligibility [27].

The same cost function as in MCTS is used, and just as in MCTS the dissipativity of the control actions is enforced by using positive actions which are multiplied with the sign of the velocity. $\alpha$ and $\lambda$ are set to 0.1 and 0.92, respectively.

### 5.2. Results

The performance of MCTS greatly depends on the number of rollouts that can be performed before the time budget is exhausted. We have observed that a single core of an Intel Xeon E5-2665 is able to perform $\sim$600 simulation steps per control step. This resulted in an average error of 2.8% of the arm length, mostly due to the large error for the far left target (see Figure 4.b-top). The error is mostly due to the limited number of simulations, as a warm start variant (which initializes the search using the chosen branch of the tree generated in the previous control step) resulted in an average error of just 0.1%. However, model discrepancies prohibit the use of such an algorithm on the real system.

When executing the controller on the robot (Figure 4.b-bottom) the error is similar to that in simulation, being 3.0%. However, instead of falling short on the far left target the system now overshoots it.

### 6. Reinforcement Learning

The previous two model-based controllers are compared to a completely model-free controller learned through temporal difference (TD) learning [27]. In a period before the actual task is to be performed, the learning algorithm interacts with the system to optimize the controller. Through trial and error it learns a feedback controller that minimizes the cost without the direct or indirect use of a model.

#### 6.1. Controller

In model-free TD learning, one tries to estimate a state-action value function $Q^*$ which indicates for each state-action pair $(s, u)$ the expected cost-to-go of taking action $u$ in state $s$ and following the control policy $\pi: s \to u$ afterwards. Because in this case only the final error is being optimized, the cost-to-go is simply the cost of the final state. In TD control the policy $\pi$ is derived from the value function by choosing the action with the lowest estimated cost-to-go. We may use the same $\epsilon$-greedy policy as in MCTS, such that

$$
\pi(s; Q) = \begin{cases} 
\min_u Q(s, u) & \text{with probability } 1 - \epsilon \\
\text{random} & \text{otherwise}
\end{cases}
$$

The optimal value function $Q^*$, defining the optimal policy $\pi^*$, is the unique solution to the Bellman equation

$$
Q^*(s, u) = c(s') + \min_{u'} Q^*(s', u')
$$

where $s'$ is the next state when taking action $u$. $Q$ is estimated by sampling state transitions $(s, u) \to (c(s'), s')$ along a trajectory and updating $Q$ towards minimizing the temporal difference error $\delta$. We use the SARSA algorithm, summarized in Algorithm 3.
In general, reinforcement learning can take quite some time before converging to an acceptable policy. However, the informative reward structure and episodic nature of the task allow for fast convergence in this case. In experiments presented here, learning took \(\sim 2\) minutes of interaction time (see Figure 5) with an average error of \(1.2\) % of the length of the arm in simulation and \(1.6\) % on the real setup. However, in both cases there was considerable variance between trials that learned to move to the same position.

The resulting trajectories (Figures 4.c) show a tendency for the simulated controller to arrive more vertically, although the observed variance prohibits firm conclusions.

7. Discussion

7.1. Comparison

The simulation and experiment results obtained show that the three different approaches to the dissipatively actuated control problem show promising results. The question now becomes: how do these approaches compare on the general case. To answer this question one must explicitly steer away from detailed quantitative analysis of the results on this test-problem, with these implementations, as such details are unlikely to transfer to other problems.

To compare the controllers qualitatively, five criteria are used: the required accuracy of the model, the planning/learning computational cost, the online computational cost, the accuracy of motion and the range of motion that the controller allows (reachability). The results of the comparison are summarized in Table 3.

As the proposed velocity-field controller depends on zero-control paths that are sensitive to model inaccuracies, it is expected that the performance is dependent on an accurate model.

So on this point, it is at a disadvantage when compared to receding horizon controllers, which are known to cope well with model inaccuracies, or reinforcement learning controllers, which require no model at all.

The cost of not using a model in reinforcement learning, is that a motion should be learned which requires a lengthy learning phase before operation can start. The velocity field controller requires only the computation of the zero-control paths, whereas the receding horizon controller requires no planning beforehand at all. But, not planning beforehand means the work itself gets harder online. The receding horizon controller requires much more online computation than the other methods. The velocity field controller requires a little computation as the correct torque needs to be computed from the velocity field. The reinforcement learning controller requires almost no computation after learning.

The dissipation constraint clearly prohibits some goal states to be reached from some initial states. But how well does the controller fill the space of feasible initial/goal state combinations? This question is particularly important as it directly relates to the robots capabilities to recover from large disturbances. The heuristic behind the velocity field controller is partially aimed at providing such reachability, but gives no guarantees. Again, for the other two controllers this reachability is a trade-off with computational costs. Because full reachability for the receding horizon controller requires the motion to be planned completely at every time step, the computational cost of reachability is expected to be lower for reinforcement learning than for the receding horizon controller.

Finally, the performance measure that was tested and compared in the experiments: accuracy of the controller. In principle, the accuracy of the velocity field controller is low, because it approaches the velocity field in infinite time, whereas the motion takes a finite time. Furthermore, there is an unknown effect caused by projecting the desired torques onto the dissipating torques. For the other two control methods, accuracy can be traded for computational costs. The robot results obtained deviate slightly from this expectation, due to a bottleneck in terms of computational cost: online computation. This bottleneck made it impossible for the receding horizon controller to simulate many variations in the control approach. This is likely the reason the receding horizon controller was less accurate than the reinforcement learning controller, and even less accurate than the velocity field controller.

![Figure 5: SARSA(\(\lambda\)) learning curves for the simulated system, averaged over 30 independent trials. The far left position takes significantly longer to converge than the other targets.](image-url)
7.2. Future work

From the comparison in table 3, obvious directions for improvement of the three controllers can be found. For the velocity field controller, the main challenge is to extend the logic behind following the no-control path into guarantees of convergence. The most promising area here seems to be to look more carefully towards the power flow within the system. The disadvantages associated with the other two controllers are more general, in the sense that they also hold for other types of problems. Some promising areas of research are using a learned model for the receding horizon controller and a more sample-efficient (batch-mode or policy search) algorithm for reinforcement learning.

8. Conclusion

This paper introduced the problem of dissipatively actuated control of mechanical systems. A type of control intuitively related to steering when skiing. The main problem in these systems is that the actuators can only brake, i.e., take energy out of the system or redistribute it. It must therefore rely on potential energy, for instance from gravity, to power the motion, and on actuation to steer. Such brake-steering can potentially be used to create robots with low energy consumption. This paper shows that standard model based non-linear control techniques do not suffice for this problem, meaning the dissipatively actuated control problem is in itself an interesting challenge. To tackle that challenge three numerically oriented approaches are proposed which have been tested on a 2-link manipulator, by making it move from one initial position to four different goal positions. In conclusion:

- **ZCVF** A newly proposed velocity field controller that is based on the heuristic of steering towards the path-of-zero-control as quickly as possible when limited control authority is available. This method requires an accurate model, but fairly limited computation both in planning and during motion. In the robot-experiments, an average accuracy of 1.9% of the length of the arm was obtained.

- **MCTS** A receding horizon control that copes with less-accurate model, but requires vast computation during motion. In robot-experiments an accuracy of 3.0% was found.

- **SARSA** A reinforcement learning technique that finds optimal solutions without using a model, but requires interaction with the real system prior to execution. In the robot-experiment the accuracy of this approach was found to be 1.2%.

Acknowledgement

W.J. Wolfslag and M.C. Plooij have received funding from the research programme STW, which is (partly) financed by the Netherlands Organisation for Scientific Research (NWO). W. Caarls has received funding from the European Community’s Seventh Framework Programme (FP7/2007-2013) under grant agreement No. 611909.


Dissipatively actuated manipulation
Dissipatively actuated manipulation

doi:10.1109/IROS.2012.6385488.

