Data-Efficient Reinforcement Learning for Energy Optimization of Power-Assisted Wheelchairs

Guoxi Feng, Lucian Bușoniu, Thierry-Marie Guerra, Sami Mohammad

Abstract—The objective of this paper is to develop a method for assisting users to push Power-Assisted Wheelchairs (PAW) in such a way that the electrical energy consumption over a predefined distance-to-go is optimal, while at the same time bringing users to a desired fatigue level. This assistive task is formulated as an optimal control problem and solved in [17] by the model-free approach Gradient Partially Observable Markov Decision Processes. To increase the data efficiency of the model-free framework, we propose here to use Policy learning by Weighting Exploration with the Returns (PoWER) with 25 controller parameters. Moreover, we provide a new near-optimality analysis of the finite-horizon fuzzy Q-iteration, which derives a model-based baseline solution to verify numerically the near-optimality of the presented model-free approaches. Simulation results show that the PoWER algorithm with the new parameterization converges to a near-optimal solution within 200 trials and possesses the adaptability to cope with changes of the human fatigue dynamics. Finally, 24 experimental trials are carried out on the PAW system, with fatigue feedback provided by the user via a joystick. The performance tends to increase gradually after learning. The results obtained demonstrate the effectiveness and the feasibility of PoWER in our application.

Index Terms—Assistive control, disabled persons, power-assisted wheelchairs, reinforcement learning.

I. INTRODUCTION

Power-assisted wheelchairs (PAWs) are becoming one of the most used tools for disabled persons in ageing societies [1]. One important characteristic of PAWs is that users can perform a tunable and suitable level of physical activities which could not be achieved with traditional manual wheelchairs or fully electric wheelchairs. Moreover, PAWs are driven by a hybrid energy source consisting of human metabolic power and electrical power from a battery. Thanks to this hybrid energy storage structure of PAWs, more degrees of freedom are available to design an optimal energy management strategy.

In this context, the major novelty of this paper is a reinforcement learning control strategy for PAWs that optimizes electrical energy while also taking into account human fatigue. We formulate the assistive task as a constrained optimal control problem: the assistive algorithm is expected to produce a desired fatigue variation of users while using minimal electrical energy for a given driving task. With the initial-to-final fatigue constraint, a (near-) optimal assistance is found so that users contribute efficiently their metabolic energy.

In contrast to hybrid electrical bicycles [2]-[4], little work is done in the PAW literature to address energy optimization with human fatigue considerations. In [5], a regenerative braking control is applied to PAWs for safe downhill driving and electrical energy savings. In [6], the control system is based on a fuzzy algorithm and the fuzzy rules are designed by an expert, aiming to increase the energy efficiency. However, human fatigue has not been taken into account to design PAWs in the literature. The adaptability of optimal solutions with respect to different human fatigue dynamics is not analyzed. An adaptable solution would be vital for PAW designs, since different users may have different fatigue dynamics. Consequently, the existing model-based approaches would not be appropriate for our PAW energy management problem.

The present study relies on Patent WO2015173094 [7] and designs assistive strategies for paraplegic wheelchair users. Specifically, we propose to use model-free reinforcement learning methods to calculate the optimal assistance while respecting a desired fatigue variation over a prescribed driving task. The optimal control method of choice is the direct policy search Policy Gradient (PG) [8]-[9]. Compared with policy iteration [10] and temporal difference learning [11], PG directly provides continuous actions without computing the value function [12], which renders it more practical in robotics [13]. Another crucial advantage of PG is its online model-free nature: it treats the wheelchair dynamics, human fatigue dynamics, and human controller as a “black box”, and the algorithm only needs state measurements and rewards (negative costs) in order to learn the solution. This is important in practice, since the true human dynamics will never be available.

Before moving on to practical implementation, the learning methodology must be evaluated in simulation with mathematical models which can roughly represent the human-wheelchair behaviors. We select the human state of fatigue (\(S_{\text{fat}}\)) model from [14], the human controller model [15] and the wheelchair model from [16] and use these models to verify
numerically the optimality of the solution found by PG. The baseline solution is given by a finite-horizon extension of the fuzzy Q-iteration in [18]. The two PG methods used in this paper are Gradient of a Partially Observable Markov Decision Processes (GPOMDP) [19]-[20] and Policy learning by Weighting Exploration with the Returns (PoWER) [21].

Compared to the previous work in [17], here we aim to improve considerably the data efficiency of the approach, by employing a different learning algorithm, PoWER, and by simplifying the parametrization of the controller. The idea is to find a near-optimal policy in much fewer trials, so as make the method better in practice. We also derive a new near-optimality analysis of fuzzy Q-iteration, which is not included in [18]. Moreover, the learning method is expected to be adaptable to either different users or changes in the same user. To verify this possibility, a novel investigation is performed in this paper. We modify the human fatigue dynamics to represent three categories of users: physically strong, normal and weak. Simulations are conducted to confirm if the proposed learning method is able to provide a solution that adapts to these cases. We also study the different convergence speeds to the baseline solution when using the parameters learned with the nominal fatigue model versus resetting them to zero defaults.

Our objective with the simulations described above is to evaluate, as a proof of concept, the effectiveness of the learning methodology in the PAW domain. To this end, we select the coarse models [14]-[16]. While these models do generate qualitatively and physically meaningful interconnected human-wheelchair behaviors [17], and thus are useful as an initial validation step, they are not required to be very accurate. Indeed, the main strength of the learning algorithm is that it does not depend on the details of the particular model or notion of fatigue used, instead working for a wide range of unknown dynamics. Having performed these simulations, our next step is to conduct an experiment with the real PAW, where the fatigue model is replaced by a joystick, using which users return a discrete subjective evaluation of their $S_{ofj}$ to the learning algorithm (too fatigued, OK, and insufficiently fatigued/desiring more exercise). This experiment serves to verify whether the learning methodology works in the real world.

Moreover, the learning method is expected to be adaptable to the human fatigue dynamics to represent three categories of users: physically strong, normal and weak. Simulations are conducted to confirm if the proposed learning method is able to provide a solution that adapts to these cases. We also study the different convergence speeds to the baseline solution when using the parameters learned with the nominal fatigue model versus resetting them to zero defaults.

This paper is organized as follows. In Section II, we present the human-wheelchair model and the problem formulation. Section III gives background on the optimal control methods applied. In Section IV, we apply PoWER and GPOMDP for PAW design and the model-free performance is compared to the model-based baseline. In Section V, we investigate the adaptability of PoWER to changes in the human fatigue dynamics. Section 0 presents the experimental results. Section 01 gives our conclusion and discusses direction for future work.

**List of symbols**

| $\tau$ | Continuous time and sampling time |
| $T_e$ | Discrete-time sample |
| $k$ | Human applied force |
| $F_m$ | Maximum available force |
| $u_m$ | Human applied torque and motor torque |
| $d_{ref}, S_{ofj-ref}$ | Desired distance-to-go and desired final state of fatigue |
| $x, u$ | State vector and control input |
| $\xi$ | Discrete-time state transition function |

$\tau$ Trajectories of states and control actions
$R$ Return
$T, r$ Terminal reward and stage reward
$K$ Finite time horizon
$Q, \hat{Q}$ Optimal $Q$-value and approximate $Q$-value
$\epsilon$ Error between $Q$ and $\hat{Q}$
$\pi^*$ Deterministic optimal control policy
$\hat{\pi}$ Model-based approximated optimal policy
$\hat{\pi}$ Model-free deterministic policy
$\hat{\pi}$ Model-free stochastic policy
$\phi$ Triangular membership function
$\bar{x}, \bar{u}$ Centre of MF $\phi_i$ and discrete action $j$
$\theta$ Model-based control parameters
$\lambda$ Model-free control parameters

**II. MODELLING AND PROBLEM STATEMENT**

Next, we introduce a human-wheelchair model which is used to validate in simulation the proposed model-free PG approaches. The proposed human model represents only coarse wheel behavior in practice, since human muscle fatigue would be difficult to precisely model or quantitatively measure [14]. However, the model is sufficiently representative to validate numerically the learning approach.

**A. Human Fatigue Model**

Owing to the repetitive nature of wheelchair pushing and the absence of a dynamical human fatigue model dedicated to PAWs in the literature, we apply the muscle fatigue model from [14] used for a cycling application. The chosen single-state human fatigue model takes into account the fatigue effect and the recovery effect which usually happen for long-term sports such as wheelchair pushing [25]. Considering these two effects, an intelligent assistance can be devised to save electrical energy. Although significant differences exist between the bicycle problems and PAW problems, this model is still qualitatively meaningful and therefore useful for numerical validation.

The dynamics of the maximum available force $F_m$ provided by human are:

$$F_m(t) = -\(\frac{F_m(t)}{M_{vc}}\) F_m(t) + R M_{vc}$$

(1)

where $0 \leq F_m(t) \leq F_m(t) \leq M_{vc}$, the variable $t$ is time, $M_{vc}$ is the Maximum Voluntary Contraction force at rest, and $F_m(t)$ is the actual human applied force. Moreover, $F$ and $R$ represent the fatigue coefficient and the recovery coefficients respectively.

When $F_{h} = F_{m}$, $F_{m}$ decreases at its maximum rate. This leads (1) to an equilibrium point where the fatigue rate is identical to the recovery rate, $F_{m} = 0$, and the positive solution is:

$$F_{eq} = \frac{R M_{vc}}{2 F} \left( -1 + \sqrt{1 + \frac{4 F}{R}} \right)$$

(2)

This positive solution $F_{eq}$ is also the minimum threshold that $F_m(t)$ can achieve. Thus $F_{eq} \leq F_{m} \leq M_{vc}$. Using the first-order Euler’s method, a discrete-time version of (1) is:

$$F_{m+1} = \left[ 1 - T_e \left( R + \frac{F_m}{M_{vc}} \right) \right] F_{m} + T_e R M_{vc}$$

(3)

with the sampling time $T_e$. Then, the state of fatigue $S_{ofj}$ in discrete time is defined as:
solution, we impose the following fatigue constraint. Knowing kinetic corresponds to a manual propulsion strategy in which all the C.

Moreover, the human force should be saturated by its suboptimality. This algorithm provides the baseline to which we will compare the existing model-free algorithms, GPOMDP [9] and Power [21]. To fully understand the model-free design, these algorithms are presented in Section B.

The system considered is described in general by the deterministic state transition function:

\[
x_{k+1} = F(x_k, u_k)
\]

where \( x \) and \( u \) are state vector and control input respectively. The general return \( R \) to optimize over a finite-horizon is:

\[
R(\tau) = \gamma^K T(x_\tau) + \sum_{k=0}^{K-1} \gamma^k r(x_k, u_k)
\]

where \( \tau = (x_0, u_0, x_1, u_1, \ldots, x_{K-1}, u_{K-1}, x_K) \) is a trajectory of the system, \( T(x_K) \) is the terminal reward, and \( r(x_k, u_k) \) is the stage reward. A discount factor \( \gamma \in [0, 1] \) may be used; in the finite-horizon case, \( \gamma \) is often taken equal to 1. The optimization problem (15) is a specific case of the general form (17). The following algorithms are presented for the general case defined in (16)-(17).

A. Finite-Horizon Fuzzy Q-Iteration

Fuzzy Q-iteration [18] is originally given in the infinite-horizon case, and the horizon-K solution can be obtained simply by iterating the algorithm \( K \) times. However, the entire time-varying solution must be maintained, and special care must be taken to properly handle the terminal reward. So for clarity we restate the entire algorithm, adapting it to the finite-horizon case.

The idea is to approximate the optimal time-varying solution, which can be expressed using Q-functions of the state in the state-space \( X \) and action in the action space \( U \). These Q-functions are generated backwards in time:

\[
Q_{k-1}(x_{k-1}, u_{k-1}) = r(x_{k-1}, u_{k-1}) + \gamma T(x_{k-1}, u_{k-1})
\]
The advantage of using $Q$-functions is that the optimal control can then be computed relatively easily, using the following time-varying state-feedback:

$$
\pi^*(x_k) = \arg \max_{u_k} Q_k^*(x_k, u_k)
$$

Since the system is nonlinear and the states and actions are continuous, in general it is impossible to compute the exact solution above. We will therefore represent $Q^*$ with an approximator that relies on an interpolation over the state space, and on a discretization of the action space. First, to handle the action, the approximate $Q$-value of the pair $(x, u)$ is replaced by that of the pair $(x, u_d)$, where $u_d$ has the closest Euclidean distance to $u$ in a discrete subset of actions $U_d = \{ \bar{u}_i | \bar{u}_j \in U, j = 1, ..., N_u \}$. To handle the state, a grid of discrete values $X_d = \{ \bar{x}_i | \bar{x}_j \in X, i = 1, ..., N_x \}$ in the state space is chosen for the centers of triangular membership functions $\phi(x) = [\phi_1(x), ..., \phi_{N_x}(x)]$ [18]. A parameter vector $\theta \in \mathbb{R}^{N_x N_u K}$ is defined, and the approximate $Q$-function is linearly interpolated by overlapping the membership functions $\phi$ on the grid of the centers $X_d$ as follows:

$$
\hat{Q}_k(x, u) = \sum_{i=1}^{N_x} \phi_i(x) \theta_{i,j,k}
$$

with $j \in \arg \min_j \| u - \bar{u}_j \|^2$. Thus, each individual parameter corresponds to a combination between a point $i$ on the state interpolation grids, a discrete action $j$, and a time stage $k$. The approximated optimal solution $\hat{\pi}$ can be obtained as follows:

$$
\hat{\pi}(x, k) = \bar{u}_j \quad \text{with} \quad j = \arg \max_j \sum_{i=1}^{N_x} \phi_i(x) \theta_{i,j,k}
$$

**Algorithm 1.** Finite-horizon fuzzy $Q$-iteration

1. for $i = 1, ..., N_x, j = 1, ..., N_u$ do
2. \[ \theta_{i,j,k-1} = r(\bar{x}_i, \bar{u}_j) + \gamma T(\bar{\pi}(\bar{x}_i, \bar{u}_j)) \]
3. end for
4. for $k = K - 2, ..., 0$ do
5. \[ \theta_{i,j,k} = r(\bar{x}_i, \bar{u}_j) + \gamma \max_{j'} \sum_{i=1}^{N_x} \phi_{i,j'}(\bar{x}_i, \bar{u}_j) \theta_{i,j',k+1} \]
6. end for
7. $\hat{\pi}(x, k) = \bar{u}_j, \quad j = \arg \max_j \sum_{i=1}^{N_x} \phi_i(x) \theta_{i,j,k} \quad \forall x, k$
8. end for

Algorithm 1 gives the complete version of Fuzzy $Q$-iteration. To understand it, note that the main update in line 6 is equivalent to the following approximate variant of the iterative update in (18):

$$
\hat{Q}_k(\bar{x}_i, \bar{u}_j) = r(\bar{x}_i, \bar{u}_j) + \gamma \max_{\bar{u}_{i,j+1}} \hat{Q}_{k+1}(\bar{x}_i, \bar{u}_{i,j+1})
$$

This is because, firstly, due to the properties of triangular basis functions the parameter $\theta_{i,j,k}$ is equal to the approximate $Q$-value $\hat{Q}_k(\bar{x}_i, \bar{u}_j)$. Secondly, the maximization over the discretized actions is done by enumeration over $j$; and thirdly, the summation is just the approximate $Q$-value at the next step, via (20). Line 2 simply sets the parameters at step $K-1$ via the initialization in (18).

For clarity, the algorithm shows in line 8 how the near-optimal control is computed via maximization over the discrete actions. In practice, this maximization is done on-demand, only for the states encountered while controlling the system, so an explicit function $\hat{\pi}$ of the continuous state does not have to be stored. Instead, only the parameters are stored.

In contrast to the algorithm itself, the infinite-horizon analysis does not easily extend to the finite-horizon case, e.g. we need to account for the possibility that $\gamma = 1$. Thus, the upcoming analysis is a novel contribution of the present paper.

Due to space limitations, we give it here without proofs, which can be found in the supplementary material at: [http://busoniu.net/files/papers/ieeetie_supp.pdf](http://busoniu.net/files/papers/ieeetie_supp.pdf).

The error $e_k$ between $\hat{Q}_k$ and $Q^*_k$ for time $k$ is defined as:

$$
e_k = \| \hat{Q}_k(x, u) - Q^*_k(x, u) \|
$$

The state resolution step $\delta_x$ is defined as the largest distance between any two neighboring triangular MF cores, i.e.

$$
\delta_x = \max_{i \in \{1, ..., N_x\}, i' \in \{1, ..., N_x\}, i \neq i'} \| x_i - x_{i'} \|
$$

The action resolution step $\delta_u$ is defined similarly for the discrete actions. Moreover, for every $x$, only $2^N_{\text{state}}$ (where $N_{\text{state}}$ is the number of states) triangular membership functions are activated. Let the infinite norm $\| \theta_k \|_\infty = \max_{i \in \{1, ..., N_x\}, j \in \{1, ..., N_u\}} \| \theta_{i,j,k} \|$ denote the largest parameter magnitude at sample $k$. Note that triangular membership functions are Lipschitz-continuous, so there exists a Lipschitz constant $L_\theta > 0$ such that $\| \phi_i(x) - \phi_i(x') \|_2 \leq L_\theta (\| x - x' \|_2) \forall x, x' \in X, \forall i$. Moreover, we say that a function of the state and action, such as the deterministic state transition function $\xi$, is Lipschitz continuous with constant $L_\xi > 0$ if $\| \xi(x, u) - \xi(x', u') \|_2 \leq L_\xi (\| x - x' \|_2 + \| u - u' \|_2) \forall x, x' \in X, u, u' \in U$.

**Assumption 1:** We assume that the reward function $r$, the terminal function $T$, and the deterministic state transition function $\xi$ are Lipschitz-continuous with the Lipschitz constants $L_r$, $L_T$, and $L_\xi$ respectively.

We present an explicit bound on the near-optimality of the $Q$-function as a function of the grid resolutions. This bound has the nice feature that it converges to zero when the grid becomes infinitely dense, which is a consistency property of the algorithm.

**Proposition 1:** Under Assumption 1, there exists an error bound $\tilde{e}_k$ so that $\hat{Q}$, i.e. the approximate $Q$-function obtained by (22) satisfies $e_k \leq \tilde{e}_k$ and $\lim_{k \to \infty} \tilde{e}_k = 0$ for $k = K - 1, ..., 0$.

Depending on the discount factor $\gamma$ and the Lipschitz constant $L_\xi$, the bound is given as follows:

1. When $\gamma < 1$, $\tilde{e}_k = (K - k) L_r (\delta_x + \delta_u) + \sum_{i=1}^{N_x} (L_T - \gamma L_r) (\delta_x + \delta_u) = (K - k) L_r (\delta_x + \delta_u) + \sum_{i=1}^{N_x} (L_T - \gamma L_r) (\delta_x + \delta_u)$
2. When $\gamma = 1$, $\tilde{e}_k = (K - k) (L_r + L_T) (\delta_x + \delta_u) = (K - k) (K - k - 1) L_r (\delta_x + \delta_u)$
3. When $\gamma > 1$, $\tilde{e}_k = (K - k) L_r (\delta_x + \delta_u) + 2 \gamma N_{\text{state}} L_\theta (\delta_x + \delta_u) \sum_{i=1}^{N_x} \| \theta_{K-x+1} \|_\infty$.
B. Policy Gradient Algorithms: GPOMDP and PoWER

In model-free policy search, exploration is indispensible to learn the unknown dynamics. Stochastic policies are needed to explore. To this end, we use a parameterized policy with the parameters \( \lambda \). Then, the stochastic policy distribution is \( \tilde{\pi}_\lambda(u_k|x_k,k) \). Under this stochastic policy, the probability distribution \( p_\lambda(\tau) \) over trajectories \( \tau \) can be expressed in the following way:

\[
p_\lambda(\tau) = p(x_0) \prod_{k=0}^{K-1} \tilde{\pi}_\lambda(u_k|x_k,k)
\]

(24)

where \( p(x_0) \) is the initial state distribution. Under trajectories \( \tau \) generated by \( \tilde{\pi}_\lambda \), the expected return is:

\[
\tilde{R}_\lambda = \int p_\lambda(\tau)R(\tau)d\tau
\]

(25)

The GPOMDP (Gradient of a Partially Observable Markov Decision Processes) algorithm [20] updates the control parameters \( \lambda \) in the steepest ascent direction so that the expected return \( \tilde{R}_\lambda \) is maximized. We apply this algorithm to estimate the gradient \( \nabla \tilde{R}_\lambda \), which can be obtained from the stage rewards \( \gamma_i \) and the distribution \( \tilde{\pi}_\lambda \). The entire procedure is given in Algorithm 2, where \( \Gamma \) is the total trials. In line 3 of Algorithm 2, for each iteration \( l \) we generate \( N_t \) trajectories using the stochastic policy with \( \lambda_l \). Applying the Likelihood Ratio Estimator, calculating the gradient \( \nabla \tilde{R}_\lambda \) is transformed to calculating \( \tilde{V}_\lambda \log \tilde{\pi}_\lambda(u_k|x_k,k) \) (more details can be found in [22], [20]). The stochastic policy distribution \( \tilde{\pi}_\lambda \) is available, so that the gradient \( \nabla \tilde{R}_\lambda \) can be computed. The expected value is approximated by Monte Carlo techniques using the \( N_t \) trajectories. The learning rate \( \alpha \) has to be tuned manually in order for the control parameters \( \lambda \) to converge efficiently.

Algorithm 2. GPOMDP

1 Initialize \( \lambda_0 \)
2 for \( l = 0, 1, 2, ... \) \( \Gamma \)
3 Generate \( N_t \) trajectories \( \tau \) of length \( K \) using \( \lambda_l \)
4 \( \tilde{V}_\lambda \tilde{R}_\lambda = \frac{1}{N_t} \sum_{k=1}^{N_t} \left[ \sum_{i=0}^{K-1} \sum_{c=0}^{C-1} \nabla \log \tilde{\pi}_\lambda(u_k^c|x_k,c,k) \right] \gamma_i \)
5 \( \lambda_{l+1} = \lambda_l + \alpha \cdot \nabla \tilde{R}_\lambda \) with the learning rate \( \alpha > 0 \)
6 end for

To obtain a higher expected return, we may consider a new distribution \( p_\lambda'(\tau) \) over trajectories that might provide a better expected return than the previous one i.e. \( \int p_\lambda'(\tau)R(\tau)d\tau \geq \int p_\lambda(\tau)R(\tau)d\tau \). The new expected return \( \int p_\lambda'(\tau)R(\tau)d\tau \) with parameters \( \lambda' \) is lower-bounded by a quantity \( L_\lambda(\lambda') \) that depends on \( \lambda \). The analytical expression of \( L_\lambda(\lambda') \) can be found in [21]. The selection of \( \lambda' \) can be done by maximizing the lower bound \( L_\lambda(\lambda') \) to implicitly maximize (25). In [23], the authors show that maximizing \( L_\lambda(\lambda') \) guarantees the improvement of the expected return. The intuition is that if \( R(\tau_1) > R(\tau_2) \), the new \( \lambda' \) will put more probability mass on \( \tau_1 \) than \( \lambda \) does.

PoWER (Policy learning by Weighting Exploration with the Returns) works by maximizing the lower bound \( L_\lambda(\lambda') \). Moreover, a deterministic policy is approximated by general basis functions \( \psi \); i.e. \( \tilde{\pi}_\lambda(x_k) = \lambda^2 \psi(x_k,k) \). To explore, Gaussian noise is added directly to the parameter vector \( \lambda \).

Using importance sampling, the parameters \( \lambda \) are updated with the \( N_s \) trials which have the highest return among the performed trials. The formula to update the parameters is [21]:

\[
\lambda_{l+1} = \lambda_l + \frac{\sum_{s=1}^{N_s}(\lambda - \lambda_l)R(\tau_s)}{\sum_{s=1}^{N_s} R(\tau_s)}
\]

(26)

The whole method is given in Algorithm 3.

Algorithm 3. PoWER

1 Initialize \( \lambda_0 \)
2 for \( l = 0, 1, 2, ... \) \( \Gamma \)
3 Generate a new trajectory \( \tau \) of length \( K \) using \( \lambda_l \)
4 Sort the performed trials decreasingly by return
5 Select the \( N_s \) trials with the highest return
6 Update parameters \( \lambda_{l+1} = \lambda_l + \frac{\sum_{s=1}^{N_s}(\lambda - \lambda_l)R(\tau_s)}{\sum_{s=1}^{N_s} R(\tau_s)} \)
7 end for

IV. DATA-EFFICIENT POLICY GRADIENT LEARNING FOR THE ASSISTANCE PROBLEM

The energy optimization problem has been solved in [17] by using a considerable amount of data, which is expensive to obtain in practice. In this section, the main purpose is to increase the data efficiency. To achieve this goal, we propose two ideas. The first one is to use a more efficient PG algorithm, namely PoWER. Secondly, as observed in [17], the operating region in the state space is concentrated on a few radial basis functions (RBFs); therefore, for the remaining RBFs the parameters remain constant or have a very small gradient. Reducing the parameters to the significant ones will accelerate the learning speed. Using Fuzzy Q-iteration as the baseline solution, we compare the performance of the two PG algorithms (PoWER and GPOMDP) with the controller parameters (29) and the one in [17]. In addition, we give an analysis of the policies obtained to explain how the assistive torques maximize the return in (15).

To represent the problem (15), the terminal reward and the stage reward of (17) are defined as follows:

\[
T(x_N) = -[w_1 w_2 \left[ (d_k - d_{ref})^2 + (S_{of_k} - S_{of-ref})^2 \right]]
\]

(27)

\[
r(x_k, u_k) = -\frac{1}{2}U_k^2
\]

(28)

where the state vector is \( x_k = [d_k, v_k, S_{of_k}]^T \) and the control input is the motor torque \( u_k = U_k \).

Since the driving task is to travel a predefined distance, negative human torque and negative motor torque are inefficient in terms of metabolic-electrical energy consumption over the driving task. Moreover, due to the actuator limitations, the maximum torque that the motor can provide is \( U_{max} \).

Therefore, the control is bounded: \( 0 \leq U \leq U_{max} \). Since the distance is monotonic, it acts as a proxy for time, which can be implicitly used by the algorithm instead of an explicit time variable. Therefore, we can use a time-invariant solution \( \tilde{\pi}_\lambda(x_k) \) to approximate the optimal time-varying solution in (19). We approximate the deterministic part \( \tilde{\pi} \) of the motor torque by the following RBF expansion:

\[
\tilde{\pi}_\lambda(x_k) = \lambda^T \psi(x_k)
\]

(29)
where the RBF \( \varphi_i = \exp(-\beta \|x_k - c_i\|^2) \), \( c_i=1\ldots M \) is the center vector of the RBFs, \( M \) is the total number of RBFs and \( \beta \) is the radial parameter. Since the radial parameter \( \beta \) is the same for each RBF, all the RBFs have the same shape. PoWER and GPOMDP for the Assistance Problem.

In the rest of the paper, for each variable, a subscript or index \( P \) (resp. \( G \)) stands for PoWER (resp. GPOMDP).

For PoWER, the general basis functions \( \psi \) in Algorithm 3 are replaced by the RBFs \( \varphi \) in (29). The exploration is carried out in the parameter space as previously explained. The zero mean Gaussian noise vector \( z_p \) with the standard deviation \( \sigma_p \) is added to the parameters and renders the action stochastic as follows:

\[
\text{sat}(0, U_{\text{max}})(\lambda^P_i + z_p)^T \varphi(x_k)
\]

where the stochastic motor torque is saturated between 0 and \( U_{\text{max}} \) and the parameter vector \( \lambda^P_i \) is updated by (26).

Regarding GPOMDP, zero mean Gaussian noise \( z_c \) is added to the executed action and renders the policy (29) stochastic. In order to prevent the executed action from violating the action saturation limits, the stochastic motor torque is selected with:

\[
q_{\text{sat}}(\lambda^G_i \varphi(x_k) + z_c)
\]

where \( q_{\text{sat}} \) is a smooth saturation (the Gaussian error function [24] shown at the top of Fig. 1) between \([0, U_{\text{max}}]\) such that the stochastic action is differentiable with respect to \( \lambda^G_i \). When the optimal action is close to the borders of the interval \([0, U_{\text{max}}]\), using the original return (28) without input saturation can lead to the divergence of the parameters. To address this problem, a penalty function \( P \) is added to the stage reward (28) as follows:

\[
r(x_k, u_k) = -\left[ \frac{1}{2} U_k^2 + w_3 P(U_k) \right]
\]

where \( w_3 \) is the constraint penalty weight. The function \( P \), shown in Fig. 1 bottom, is defined as follows:

\[
P = \begin{cases} 
\sin\left(\pi \frac{(U - U_{\text{max}})}{0.04 U_{\text{max}}}\right) + 1 & 0.98 U_{\text{max}} \leq U \leq U_{\text{max}} \\
0 & 0.02 U_{\text{max}} \leq U \leq 0.98 U_{\text{max}} \\
\sin\left(\pi \frac{-(U)}{0.04 U_{\text{max}}}\right) + 1 & 0 \leq U \leq 0.02 U_{\text{max}}
\end{cases}
\]

which penalizes the (stochastic) action when it is close to the saturation value. The objective of \( P \) is to keep the mean value of the stochastic actions inside the interval \([0, U_{\text{max}}]\).

Recall that we use a time-invariant policy. Consequently, the stochastic action distribution does not depend on the time stage \( k \), but on the state \( x_k \). According to (31), the distribution \( \hat{\pi}^G_i(U_k | x_k) \) of the stochastic motor torque \( U \) is:

\[
\hat{\pi}^G_i(U_k | x_k) = \frac{1}{\sqrt{2 \pi \sigma^2}} \exp\left(-\frac{q_{\text{sat}}^{-1}(U_k) - \lambda^G_i \varphi(x_k)}{2 \sigma^2} \right)
\]

The derivative of (34) with respect to \( \lambda^G_i \) is used to estimate the gradient \( \nabla_k R_k \) in Algorithm 2 and to update the parameter vector \( \lambda^G_i \).

By tuning the parameters \( (\beta, c, M) \) of the basis functions (29), the standard deviation \( \sigma_p \) and \( \sigma_c \), the reward weights \( (w_1, w_2) \), the learning rate \( \alpha \) and the penalty weight \( w_3 \), we have all the conditions to update the parameters \( \lambda^P \) or \( \lambda^G \).

A. Results with PoWER and GPOMDP

In this section, simulations are carried out to compare the proposed methods. The whole set of parameters is shown in Table I. The human model parameters are adapted from [14] to have a reasonable fatigue and recovery rate, which render the optimal more challenging and avoid a trivial optimal solution. The control strategy is approximated over the state-space and action-space region given in Table I. The configurations and learning parameters of the return function, penalty function, model-based policy, and model-free policies are shown in Table II later.

For the simulation in Fig. 2, the exploration noise (and the learning rate for GPOMDP) are tuned to deliver a good performance for each configuration and each algorithm. The number in the legend gives the total number of controller approximation (29) for each simulation. A mean value along with a 95% confidence interval calculated for 10 independent simulations is given (each simulation with 400 trials). Fig. 2 shows that with the same policy parametrization, PoWER has a considerably higher data efficiency than GPOMDP. GPOMDP-25 and GPOMDP-200 give a similar final performance. Considering the mean, 90% of the baseline return is provided in around 100 trials by PoWER-25. The same performance is...
PoWER and fuzzy Q-iteration respectively. Here again, PoWER delivers a better solution than GPOMDP in terms of final return.

The simulation was done on an Intel Core i7-6500 CPU at 2.50GHz. The average elapsed CPU time to compute a control action is $1.0007 \times 10^{-4}$ s, $7.6464 \times 10^{-4}$ s, $1.4562 \times 10^{-4}$ s, and $7.5934 \times 10^{-4}$ s respectively for PoWER-25, PoWER-200 GPOMDP-25 and GPOMDP-200. As their elapsed CPU time is significantly less than the sampling time of 0.05s, it is possible to embed them into a real PAW.

**Remark 2:** The RBF grid used for policy approximation has only one center on the $S_{of}$ state for the policy approximation. It may appear surprising that the controller works despite this limitation. Indeed, we have tested it and it works well for initial $S_{of}$ in the interval [0.4, 0.6]. This is because the solutions are similar for this range of $S_{of}$, as confirmed by the baseline fuzzy Q-iteration. In additional PG experiments with three centers on the $S_{of}$ state and reduced radius in this dimension, performance increased marginally but at the cost of less reliable convergence. Another possible reason is that much of the optimal control input may be open-loop, and again since the distance is monotonic, it acts as a proxy for time. Even if this were so, state feedback is nevertheless crucial in practice to defend against disturbances, so we choose to apply closed-loop, state-feedback policies.

In the beginning, the motor provides a large assistance and the user pushes “hard” to start the wheelchair. After reaching a suitable velocity, the user reduces his applied force to recover. In the remainder of the task, the motor assistance is reduced gradually to minimize the energy consumption while the user continues recovering. The assistance provided tries to use as little electrical energy as possible and enables the user to reach the desired final fatigue state.

PoWER and GPOMDP have a similar terminal error of nearly 0.05 between the final $S_{of_K}$ and the desired final value.

---

**TABLE I**

<table>
<thead>
<tr>
<th>Meaning</th>
<th>Notation</th>
<th>Value or domain</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sampling time</td>
<td>$T_s$ [s]</td>
<td>0.05</td>
</tr>
<tr>
<td>Human parameters</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Recovery coefficient</td>
<td>$R$ [s$^{-1}$]</td>
<td>0.0063</td>
</tr>
<tr>
<td>Fatigue coefficient</td>
<td>$F$ [s$^{-1}$]</td>
<td>0.153</td>
</tr>
<tr>
<td>MVC</td>
<td>$M_{ve}$ [N]</td>
<td>100</td>
</tr>
<tr>
<td>Fraction of $V_{max}$</td>
<td>$f$</td>
<td>0.5</td>
</tr>
<tr>
<td>Human control gain</td>
<td>$K_p$</td>
<td>30</td>
</tr>
<tr>
<td>Wheelchair parameters</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wheel radius</td>
<td>$\zeta$ [m]</td>
<td>0.33</td>
</tr>
<tr>
<td>Maximum velocity</td>
<td>$V_{max}$ [rad/s]</td>
<td>7</td>
</tr>
<tr>
<td>System matrix</td>
<td>$A$</td>
<td>$\begin{bmatrix} 1 &amp; 0.05 \ 0.9406 &amp; 0 \end{bmatrix}$</td>
</tr>
<tr>
<td>Input matrix</td>
<td>$B$</td>
<td>$\begin{bmatrix} 0 \ 0.0059 \end{bmatrix}$</td>
</tr>
<tr>
<td>Driving schedule</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Finite horizon</td>
<td>$K$</td>
<td>200</td>
</tr>
<tr>
<td>Initial state of fatigue</td>
<td>$S_{of_a}$</td>
<td>0.5</td>
</tr>
<tr>
<td>Desired final human fatigue</td>
<td>$S_{of_{goal}}$</td>
<td>0.5</td>
</tr>
<tr>
<td>Distance-to-go</td>
<td>$d_{ref}$ [rad]</td>
<td>20</td>
</tr>
<tr>
<td>State-space and action-space region</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Distance</td>
<td>$d$ [rad]</td>
<td>[0.20]</td>
</tr>
<tr>
<td>Velocity</td>
<td>$v$ [rad/s]</td>
<td>[0.7]</td>
</tr>
<tr>
<td>State of fatigue</td>
<td>$S_{of}$</td>
<td>[0.35, 0.7]</td>
</tr>
<tr>
<td>Motor torque</td>
<td>$U$ [Nm]</td>
<td>[0.50] ($U_{max} = 50$)</td>
</tr>
</tbody>
</table>

---

**TABLE II**

<table>
<thead>
<tr>
<th>Return function and penalty function configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reward weight matrix $[W_1 \ W_2]$</td>
</tr>
<tr>
<td>Penalty weight $w_a$</td>
</tr>
<tr>
<td>Q-function approximation</td>
</tr>
<tr>
<td>Centers of triangular functions $\phi$</td>
</tr>
<tr>
<td>Number of equidistant discrete actions</td>
</tr>
</tbody>
</table>

**Radial basis functions (29) configuration 1**

| Radial parameter $\beta$ | 0.5 |
| Centers of RBF's distributed on an equidistant grid | $5 \times 5 \times 8$ |
| Total number of RBF's $M$ | 200 |

**Radial basis functions (29) configuration 2**

| Radial parameter $\beta$ | 0.5 |
| Centers of RBF's distributed on an equidistant grid | $5 \times 5 \times 1$ |
| Total number of RBF's $M$ | 25 |

**GPOMDP parameters**

| Learning rate $\alpha$ | $10^{-3}$ |
| Standard deviation $\sigma_e$ | 5 |

**PoWER parameters**

| Importance sampling $N_r$ | 10 |
| Standard deviation $\sigma_p$ | 1 |
In order to verify whether the assistive control can adapt to a bigger range of parameter changes, we carry out the same comparison for \( \eta = 8, 4, 3, 1/3, 1/4, 1/8 \). Table III gives the baseline return for each \( \eta \), the minimal return for each case and the number of trials to converge to 90% of the corresponding baseline return for both initializations. The asterisk * represents situations where the learning algorithm fails to converge to 90% of the baseline return within 400 trials.

Table III shows that both initializations have similar convergence for \( \eta = 8 \). For \( \eta = 3 \), the initialization to zero has a faster convergence. This result may be because that the initialization to zero is closer to the optimal solution. Nevertheless, for all the other \( \eta \), the initialization with the nominal model converges faster. Overall, starting learning with the nominal solution can guarantee a higher minimum return. Moreover, PoWER with prior knowledge adapts reasonably well to human fatigue dynamics changes without tuning again the learning parameter \( \sigma \). This study therefore confirms the possibility of providing an adaptive solution for different human fatigue dynamics.

### VI. Real-Time Experiment

To demonstrate the effectiveness of the proposed learning algorithm, proof-of-concept experiments have been conducted on our PAW prototype, which is equipped with two torque sensors, two position encoders and a joystick. Via the joystick, the user can return a subjective evaluation of their \( S_{of-ref} \) to the control algorithm. When the user pushes the joystick to the negative or positive Y-direction, the joystick returns to the algorithm a discrete value \(-1 \) or \( 1 \), respectively. The neutral position of the joystick returns a discrete value \( 0 \). These three discrete values \(-1, 0 \) and \( 1 \) mean respectively that the user feels too tired, is comfortable, and feels insufficiently tired (is willing to exercise more). The discrete signal is filtered so that when it changes between two levels (among \(-1, 0 \) and \( 1 \)), its filtered version \( f \) provides a gradual transition between these levels. Furthermore, to avoid the need for too many pushes of the joystick, after such a transition the filtered signal is kept nearly constant for a certain duration.
The driving scenario consists in riding on a straight flat road with a given reference velocity $v_{ref}$ set by the user. The velocity estimated from the position encoders is available via the computer connected to the data acquisition system. The control objective is to minimize both the electrical energy and the use of the joystick, while tracking the reference velocity. Therefore, the stage reward function is:

$$r_k = -w_1 \left(v_k - v_{ref_k}\right)^2 - w_2 I_k^2 - w_3 U_k^2$$  \hspace{1cm} (35)

where $v_{ref_k}$ is the given reference velocity at discrete sample $k$. The reward weights are $w_1=10$, $w_2=0.25$, and $w_3=0.05$. Note that any joystick signal $I \neq 0$ is penalized. The controller is configured as a PI-type law:

$$U_k = \lambda_1 \left(v_k - v_{ref_k}\right) + \lambda_2 \sum_{i=0}^{k} (v_i - v_{ref_i}) + \lambda_3 I_k$$  \hspace{1cm} (36)

The first four terms of the controller (36) are used to track the reference values $v_{ref}$ whilst keeping the filtered joystick signal $I$ to 0. The term $\lambda_3 F_{h_k} \zeta$ is for compensating the human input.

One healthy male volunteer (29-year-old) performed the proof-of-concept experiments. There are 5-minute rest periods between consecutive trials. In total, 24 trials with the same driving condition have been carried out on the same day to learn the parameter vector $\lambda$ in (35). Fig. 5 shows the total return of each trial. Among the 24 trials, 3 trials went unstable at the beginning of learning. For these trials, the user stopped immediately the wheelchair and a very low return was given to the learning algorithm to avoid such situations in the future. The return tends to increase gradually after performing these trials.

We notice that the obtained curve of return is noisy. Due to the time-consuming nature of the experiment, it is not feasible to perform many trials to obtain a smooth mean return. Therefore, we analyze qualitatively the obtained trajectories.

Fig. 6 shows the trajectories of the first four stable trials and the last four trials. We remark that the user does not push the joystick anymore in the last four trials. The joystick signal $I$ sums up the influence of main physiological and psychological factors to tell the learning algorithm what assistive torque is suitable to users. The fact that the user does not use anymore the joystick means that after training, the provided assistive torques are acceptable in terms of the sensation of fatigue. Another consequence of training is that the user and the controller track together the given velocity more smoothly.

Through these proof-of-concept experiments, we conclude that the proposed learning algorithm PoWER is able to improve the performance of the controller (36). For a final commercial product, there will be a certain accommodation time to obtain a satisfactory performance, during which a health professional would help the user interact with the PAW.

VII. CONCLUSION

In this paper, a novel PAW control design has been proposed for paraplegic wheelchair users. The assistive strategy is based on energy optimization, while maintaining a suitable fatigue level for users and using minimal electrical energy over a distance-to-go. This optimal control problem was solved by the online model-free reinforcement learning methods PoWER and GPOMDP. Their near-optimality was confirmed by the model-based approach finite-horizon Q-fuzzy iteration. An important contribution is that the near-optimality of finite-horizon Q-
fuzzy iteration was proven. In addition, simulation results confirmed that PoWNER with a simplified controller parameterization provides a considerably higher data efficiency, which makes the model-free framework better applicable in practice. Moreover, an investigation has been done to illustrate that PoWNER is also able to adapt to human fatigue dynamics changes. Finally, a proof-of-concept experiment has been carried out to demonstrate the feasibility of the approach in practice. Future work will focus on validating the adaptability of the applied assistive algorithm for different users with the real wheelchair.

REFERENCES


Guoxi FENG received the M.Sc. degree in control engineering from the Université Polytechnique Hauts-de-France (UPHF), Valenciennes, France, in 2016. He is currently a PhD candidate at the UPHF. His current research interests include reinforcement learning and observer design for power-assisted wheelchair applications.

Lucian Buşoniu received the M.Sc. degree (valedictorian) from the Technical University of Cluj-Napoca, Romania, in 2003, and the Ph.D. degree (cum laude) from the Delft University of Technology, the Netherlands, in 2009. He is a Full Professor with the Department of Automation at the Technical University of Cluj-Napoca, where he leads the group on Robotics and Nonlinear Control. He has previously held research positions in the Netherlands and France. His research interests include nonlinear optimal control, reinforcement learning and approximate dynamic programming, multiagent systems, and robotics. He received the 2009 Andrew P. Sage Award for the best paper in the IEEE Transactions on Systems, Man, and Cybernetics.

Thierry-Marie Guerra received the Ph.D. degree in automatic control from the Université Polytechnique Hauts-de-France (UPHF), France, in 1991 and the HDR degree in 1999. He is currently a Full Professor at the UPHF and Head of the CNRS Laboratory LAMIH http://www.univ-valenciennes.fr/LAMIH/. His current research interests include wine, hard rock, chess, nonlinear control, LPV, quasi-LPV (Takagi–Sugeno) models control and observation, nonquadratic Lyapunov functions and their applications to power train systems (IC engine, hybrid vehicles) and to disabled people. He is the Chair of IFAC TC 3.2 “Computational Intelligence in Control”, member of the IFAC TC 7.1 Automotive Control, and Area Editor of the international journals Fuzzy Sets and Systems and IEEE Transactions on Vehicular Technology.

Sami Mohammad received the Ph. D. degree in automatic control from the university of Valenciennes and Hainaut-Cambrésis (UVHC), France, in 2011. He is now the CEO of Autonomad Mobility (www.autonomad-mobility.com), an innovative start-up specialized in high added-value mobility aids for disabled people. His research is focused on applied smart electrical mobility aids using advanced automatic control theories.