Expert Systems with Applications 42 (2015) 2234–2255

Contents lists available at ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

On-line policy learning and adaptation for real-time personalization of an artificial pancreas

Mariano De Paula [a], Gerardo G. Acosta [b], Ernesto C. Martínez [c,*]

[a] INTELYMEC – Centro de Investigaciones en Física e Ingeniería del Centro – CIFICEN – CONICET, Av. del Valle 5737, Olavarría B7400JWI, Argentina
[b] UNCPBA-Facultad de Ingeniería, Universidad Nacional del Centro de la Provincia de Buenos Aires, Av. del Valle 5737, Olavarría B7400JWI, Argentina
[c] INGAR (CONICET-UTN), Avellaneda 3657, Santa Fe S3002 GJC, Argentina

ARTICLE INFO

Article history:
Available online 31 October 2014

Keywords:
Diabetes
Gaussian processes
Glycemic variability
On-line sparsification
Policy learning
Reinforcement learning

ABSTRACT

The dynamic complexity of the glucose–insulin metabolism in diabetic patients is the main obstacle towards widespread use of an artificial pancreas. The significant level of subject-specific glycemic variability requires continuously adapting the control policy to successfully face daily changes in patient's metabolism and lifestyle. In this paper, an on-line selective reinforcement learning algorithm that enables real-time adaptation of a control policy based on ongoing interactions with the patient so as to tailor the artificial pancreas is proposed. Adaptation includes two online procedures: on-line sparsification and parameter updating of the Gaussian process used to approximate the control policy. With the proposed sparsification method, the support data dictionary for on-line learning is modified by checking if in the arriving data stream there exists novel information to be added to the dictionary in order to personalize the policy. Results obtained *in silico* experiments demonstrate that on-line policy learning is both safe and efficient for maintaining blood glucose variability within the normoglycemic range.

1. Introduction

Diabetes is a chronic disease due to the inability of the pancreas to segregate the insulin and glucagon hormones, which gives rise to anomalous blood glucose levels in individuals and mainly alter their carbohydrate, fat and protein metabolisms. High level of blood glucose concentration causes considerable long-term complications such as cellular dehydration, tissue damage, vascular injury that increases the risks to heart attacks, cardiovascular diseases, strokes, end-stage renal diseases and blindness (Guyton & Hall, 2000; Siegelaar, Holleman, Hoekstra, & DeVries, 2010). To overcome these complications is necessary to maintain the blood glucose level as near to the normal levels as possible. Through continuous insulin infusion by an external pump it is possible to reduce glycemic variability and prevent events of hyper- or hypoglycemia (Hanaire et al., 2008). Recent advances in technological devices like glucose sensors and insulin delivery pumps allow moving closer towards the artificial pancreas (AP) (Cobelli, Renard, & Kovatchev, 2011), which includes such devices as key hardware elements in the glucose regulation loop. The other essential component of an artificial pancreas is the control strategy used to calculate the insulin delivery rate upon a glucose measurement signal coming from the sensor (Lunze, Singh, Walter, Brendel, & Leonhardt, 2013).

The major challenge for the development of an artificial pancreas is to cope with the erratic changes in blood glucose levels due to the variability of glucose metabolism between patients and for the same person over time expressed in terms of age, insulin sensitivity, life style, exercise routine, feeding habits, etc (Bequette, 2012; Buckingham, Caswell, & Wilson, 2007; Heinemann, 2002; Krüger, Slabber, Joubert, Venter, & Vorster, 2007). Moreover, despite important developments in sensor and insulin pump technology, the control system of an AP must cope with delays and inaccuracies in both glucose sensing and insulin delivery (control action). In the past three decades the most extensively used control strategies in the design of control algorithms for an artificial pancreas were the proportional–integral-derivative (PID) and the model predictive control (MPC) (Peyser, Dassau, Breton, & Skyler, 2014). Also, fuzzy logic control techniques were developed but to a lesser extent. The main obstacle for most control algorithms is that unpredictable factors, such stress and illness, are ubiquitous sources of glycemic variability that must be handled properly. Most of the existing control literature have failed to address the challenge of patient-specific variability mainly because a fixed control policy is applied bearing in mind an "average" patient. Recent research efforts were focused towards adaptive control techniques to develop an artificial

* Corresponding author.
E-mail addresses: marianodepaula@gmail.com (M. De Paula), gerardo.acosta@ieee.org (G.G. Acosta), ecmarti@santafe-conicet.gob.ar, ecmarti@santafe-conicet.gov.ar (E.C. Martínez).

http://dx.doi.org/10.1016/j.eswa.2014.10.038
0957-4174/

pancreas which is capable to deal with the uncertain behavior in diabetic patients (Turksoy & Cinar, 2014).

The well known proportional–integrative–derivative feedback controller was employed as an early close-loop control algorithm for the AP (Bequette, 2005). Several versions of the PID controller have been developed and evaluated *in silico*. Chee, Fernando, Savkin, and van Heeden (2003) designed a PID controller applying the concept of an expert system, achieving a "sliding table" to implement a PID control scheme with dynamic properties. A PID is a purely reactive controller, i.e. responds to changes in glucose concentration after they have occurred. To improve PID performance, Weinzimer et al. (2008) add a feed-forward action and Marchetti, Barolo, Jovanovic, Zisser, and Seborg (2008), incorporate switching strategies for initialization after a meal and insulin bolus, together with time-dependent trajectories and filters for noise removal in continuous glucose monitoring. Gantt, Rochelle, and Gatzke (2007) designed a simpler asymmetric PI controller with adjustable parameters which are tuned by local search methods. As a result, the PI controller may exhibit a different response depending on the sign of the observed error value. To deal with the strong nonlinearity behavior of glucose insulin dynamics in Sasi and Elmalki (2013), a PID and slider table controller are proposed to address the glucose disturbance caused by exercise, delay or noise in glucose sensor and nutrition mixed meal absorption at meal times.

Model predictive control (MPC) is a model-based control technique (García, Prett, & Morari, 1989), which was extensively adopted for simulation-based evaluation of glucose regulation in Type 1 diabetic patients. In the work of Magni et al. (2007), a linear MPC controller was assessed whereas in Magni et al. (2009) the trial was extended to evaluate a nonlinear state feedback MPC controller (Hovorka et al., 2004). In Lee, Buckingham, Wilson, and Bequette (2009) a feedback control algorithm using MPC along with meal detection and meal size estimation approach are combined. In addition, a pharmaco-dynamic model of insulin action is used to provide "insulin-on-board" constraints that explicitly include the future effect of past and current changes to the insulin delivered. Furthermore, a pump shut-off feature is included to avoid hypoglycemic events. In order to reduce the computational effort and explicitly deriving the control policy the work of Dua, Doyle, and Pistikopoulos (2006) proposes the integration of the parametric programming optimization technique with the MPC framework to obtain the control actions (changes to the insulin delivery rate) as a function of the current blood glucose concentration of the patient (state variables) by treating the control actions as optimization variables and the state variables as inputs. The multi-parametric model-based control (mp-MPC) is an evolved version, which is a control method with the ability to solve the on-line optimization problem, involved in traditional MPC, off-line via parametric optimization with the advantage that control actions are calculated in an on-line way via simple function evaluations instead of solving repetitively on-line a computationally demanding optimization problem (Dua, Kouramas, Dua, & Pistikopoulos, 2008; Percival et al., 2011). Alternatively, a zone-MPC has been developed (Grosman, Dassau, Zisser, Jovanovic, & Doyle, 2010) which uses mapped-input data and is adjusted automatically by linear difference personalized models, being the target of the control variable a zone instead of a set point or trajectory. In Favero et al. (2014), a modular MPC algorithm has been developed based on a meal-informed MPC strategy and it was first used in an outpatient wearable AP.

Determination of PID constants through well-known methods, such as the well known Ziegler–Nichols, is difficult since a well-tuned AP is always a patient-specific issue. Also, changes in glucose dynamics generate the need for constantly updating these parameters. On the other hand, MPCs are model-based control strategies that have an important drawback for using them in an AP: the controller performance is strongly dependent on the accuracy of the model used to represent the true glucose–insulin dynamics. To overcome these drawbacks, adaptive control techniques arise as a promising alternative to address glycemic variability in diabetic patients (Turksoy & Cinar, 2014). An adaptive control system that do not necessitate any meal or activity announcements was proposed in Turksoy, Bayrak, Quinn, Littlejohn, and Cinar (2013), which is mainly based on the generalized predictive control framework. In Eren-Oruklu, Cinar, Quinn, and Smith (2009) a lag filter is used to account for the time-lag between subcutaneous and blood glucose values. Also, a Smith predictor is used to cope with delays in insulin action effects, which provides a model based-control strategy that calculates the required insulin infusion rate while model parameters are recursively tuned. In a similar research avenue, machine learning techniques have been applied in the attempt for integrating adaptive control strategies in glucose regulation. In the pioneering work of McCausland, Mareels, Barnett, and Arad (1999) a rule-based algorithm was proposed, which starts from a set of generic rules to form conclusions. Later on, decision rules are adapted to suit a given person characteristics by learning from patient-specific data. In Cosenza (2012) fuzzy techniques are employed for the development of a decision support system which operates as a off-line mixed feedback–feedforward controller allowing the optimization of postprandial glycemia in type 1 diabetic patients. In De Paula and Martínez (2012a), a simulation based approach based on Gaussian process dynamic programming (Deisenroth, Rasmussen, & Peters, 2009) is implemented to find a control policy, which is represented in a compact format that provides plenty of room for its personalization. Run to run algorithms have been used for fine tuning of basal infusion rates using sparse blood glucose measurements (Palerm, Zisser, Jovanovič, & Doyle, 2008).

A machine learning algorithm can be considered as a systematic procedure for extracting structure from data (Deisenroth, 2010). In particular, the work of Esfandiari, Babavalian, Moghadam, and Tabar (2014) review a set of techniques to extract knowledge from the measured patient data. Automatically learning from the patient data is of paramount importance to develop true adaptive control strategies. In Serhani, Benharref, and Nujum (2014), an adaptive Expert System is proposed that implements an iterative technique based on previous experience to increasingly improve clinical-decision making. In Akbari Torkestani and Ghanaat Pisheh (2014), a learning automata-based mechanism is proposed to determine the insulin doses taking into account the past history of blood glucose measurements which, in turn, are encoded in the parameters of a Gaussian distribution function. In the work of Daskalaki, Prountzou, Diem, and Mougiakakou (2012), continuous glucose monitoring is used to develop on-line adaptive data-driven models for glucose prediction to account for the glycemic variability of diabetic patients. An integration of reinforcement learning (RL) (Sutton & Barto, 1998) and optimal control theory have been presented in Elena Daskalaki, Diem, and Mougiakakou (2013), where a model-free control approach is proposed. The control strategy admits an initialization based on clinical procedures, includes simultaneous adjustment of both the insulin basal rate and the bolus dose, and makes room for a real-time personalization of the control policy. In comparison with other traditional control strategies, RL does not require a detailed description of the glucose dynamics in terms of a first-principles model. Learning algorithms developed as model-based control strategies are faster than those based on model-free strategies. RL algorithms aim to find a policy control that optimizes a long-term performance measure, which makes it ideal to cope with delayed effects of the delivered insulin since that effect will be taken into account by the state-value function or state-action function (Daskalaki et al., 2013). For RL algorithms to be practical, they must be able to learn from a handful of samples

while continually taking actions in real-time (Hester, Quinlan, & Stone, 2012). The paradigm of model-based learning strategies, borrowed from the field of the robotics, require a model of the dynamics (in our case a model of glucose–insulin dynamics) to describe essential information about the response of the system to control actions (Nguyen-Tuong & Peters, 2011b). Non-parametric models are true data models suitable to work with erratic glucose–insulin dynamics (Daskalaki et al., 2013, 2012), which makes them a powerful tool for adaptive control by reducing the problem of model bias, which frequently occurs when deterministic models are used. In (Ruan, Thabit, Kumareswaran, & Hovorka, 2014), the proposal is to estimate the parameters of the model developed to describe the insulin pharmaco-kinetics; a Bayesian inference approach is used based on the collected data from patients. More specifically, a Gaussian process is a novel modeling tool within the non-parametric framework which when combined with tractable Bayesian inference (Rasmussen & Williams, 2006) is useful to cope with modeling the glucose–insulin dynamics under uncertainty (Markakis, Mitsis, Papavassilopoulos, & Marmarelis, 2010). For safety and performance in the artificial pancreas, on-line adaptive strategy requires to work in real-time and with continuous data streams. Therefore, for a patient model to be useful in controlling glycemic variability it should be incrementally updated in real-time (Gijsberts & Metta, 2013). Recently proposed on-line sparsification techniques (Chen, Gao, & Wang, 2013; Engel, Mannor, & Meir, 2004, NguyenTuong and Peters, 2011a) are best suited for on-line learning of dynamic models that adapt a control policy to a patient lifestyle and unique glucose metabolism.

Model-based RL is a promising alternative to develop adaptive personalization techniques to control blood glucose since RL algorithms can quantitatively predict and measure the performance of selected actions (Bothe et al., 2013; Daskalaki, Diem, & Mougiakakou, 2013). The glucose–insulin dynamics has a continuum of states and actions. Therefore, a truly flexible and adaptive proposal based on the model-based RL framework should be made in a continuous representation. In RL, to deal with this situation an efficient balance between exploitation and exploration to generalize from experience is needed. Also, adaptive modeling of the dynamics is critical. In this paper, an on-line selective reinforcement learning algorithm that enables real-time adaptation of a control policy based on ongoing interactions with the patient so as to tailor the artificial pancreas is proposed. Adaptation includes two online procedures: on-line sparsification and parameter updating of the Gaussian process used to approximate the control policy. Probabilistic Gaussian process models of the transition dynamics and value functions are built on-line. Also, the control policy itself is modeled by a Gaussian process model. For safety and performance, the policy is adapted using only a small number of interactions with the patient and only relevant data is sampled through Bayesian active learning (Baranes & Oudeyer, 2013; Cohn, Ghahramani, & Jordan, 1996; Deisenroth et al., 2009). The integration of a sparsification technique with the on-line learning algorithm makes possible to work with a continuous data stream which enables a permanent updating of the support data set used to approximate the control policy. In this way, on-line adaptation of the control policy is effective for controlling subject-specific variability due to a patient's lifestyle and its distinctive metabolic response. Illustrative examples are used to discuss implementation details of the proposed approach.

2. Off-line policy learning

The notion of a *control policy* comes from the field of artificial intelligence (Russell & Norvig, 2009), and is especially used to describe the behavior of an intelligent agent (controller) in the field of RL (Sutton & Barto, 1998). In mathematical terms, the policy is a function π:$\mathbf{x} \rightarrow \mathrm{u}$ that maps states ($\mathbf{x}$) to a control action (u). In this sense, a control policy defines how a controller chooses the control actions to be applied to the controlled system. More specifically, in the artificial pancreas the control policy determines the insulin infusion rate which should be delivered to a diabetic patient. By formulating and solving a RL problem it is possible to find a *generic* policy for controlling blood glucose by simulating uncertain conditions in a *generic* patient and its environment which give rise to glycemic variability. Later on, the generic policy can be personalized in real-time to a given patient.

2.1. Reinforcement learning

Solving a RL problem consists in learning iteratively a task from interactions to achieve a goal. During learning, an agent (or controller) interacts with the target system by taking an action $\mathrm{u}_t \in \mathbb{U} \subseteq \mathbb{R}$ and, after that, the system evolves from the state $\mathbf{x}_t \in \mathbb{X} \subseteq \mathbb{R}^{n_x}$ to $\mathbf{x}_{t+1}$ and the agent receives a numerical signal r_t called *reward* (or cost) which provides a measure of how good (or bad) is the action taken at $\mathbf{x}_t$ in terms of the observed state transition. Rewards are given as hints regarding goal achievement or optimal behavior. In applying RL to the artificial pancreas, the main objective of the agent is to learn the optimal policy, π^*, which defines the optimal insulin infusion for different patient's states, bearing in mind both short and long term rewards. Let us assume that under a given policy π, the expected cumulative reward $V^{\pi}(\mathbf{x})$, or value function over a certain time interval, is a function of $\mathbf{x}^{\pi}$, where $\mathbf{x}^{\pi} = \{\mathbf{x}_t\}_{t=1}^{t=N}$ are the corresponding state values and $\mathbf{u}^{\pi} = \{\mathrm{u}_t\}_{t=1}^{t=N}$ defines the policy-specific sequence of control actions. The sequence $\mathbf{x}^{\pi}$ of state transitions gives rise to rewards $\{r_t\}_{t=1}^{t=N}$. Blood glucose regulation is a continuous task without a single final state, hence N could be defined to account for the daily routine including meals, exercise, sleep, etc. Therefore, the discounted sum of future rewards $R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \ldots = \sum_{k=o}^{\infty} \gamma^k r_{t+k+1}$ is used to define the (discounted) expected state-value function for a policy π from the state $\mathbf{x}$:

$$V^{\pi}(\mathbf{x}) = E_{\pi}\{R_t|\mathbf{x}_t = \mathbf{x}\} = E_{\pi}\left\{\sum_{k=o}^{N} \gamma^k r_{t+k+1}|\mathbf{x}_t = \mathbf{x}\right\} \quad (1)$$

where $\gamma \in (0,1]$ is the discount factor which weights future rewards. $V^*(\mathbf{x})$ is used to denote the maximum discounted reward obtained when the agent starts in state $\mathbf{x}$ and executes the optimal policy π^*. Thus, the associated optimal state-value function satisfies the Bellman's equation for all state $\mathbf{x}$ is:

$$V^*(\mathbf{x}_t) = \mathrm{argmax}_{\mathrm{u}}\{r_t + \gamma . E_{\mathbf{x}_{t+1}}[(V^*(\mathbf{x}_{t+1})|\mathbf{x}_t, \mathrm{u}_t)]\} \quad (2)$$

where $\mathrm{u}_t = \pi^*(\mathbf{x}_t)$. Similarly, the state–action value function Q^* is defined by:

$$Q^*(\mathbf{x}_t, \mathrm{u}_t) = r_t + \gamma . E_{\mathbf{x}_{t+1}}[(V^*(\mathbf{x}_{t+1})|\mathbf{x}_t, \mathrm{u}_t)] \quad (3)$$

such that $V^*(\mathbf{x}) = \mathrm{argmax}_{\mathrm{u}} Q^*(\mathbf{x}, \mathrm{u})$ for all $\mathbf{x}$. Once Q^* is known through interactions, then the optimal policy can be obtained directly through:

$$\pi^*(\mathbf{x}) = \mathrm{argmax}_{\mathrm{u}} Q^*(\mathbf{x}, \mathrm{u}) \quad (4)$$

When the state space $\mathbb{X}$ and the action space $\mathbb{U}$ are discrete and finite, the optimal policy can be obtained directly from a tabular representation of the value function. However, in our problem the state and actions take their values from a continuum. Hence, a function approximation technique is required for value function approximation. Moreover, inductive modeling of the optimal policy is mandatory to allow selecting optimal actions in the continuous space $\mathbb{U}$. Gaussian process models are a powerful alternative for generalization and on-line learning in reinforcement learning algorithms.

2.2. Gaussian process

Gaussian process (GP) is a relatively new kernel method popularized within the machine learning community by Rasmussen and Williams (2006). It is a powerful, non-parametric tool for regression in high dimensional spaces returning a non-parametric probabilistic model. A GP is a generalization of a Gaussian probability distribution where the distribution is over functions instead of assuming a model with a given structure before data is considered.

A key advantage of the GP is the ability to provide uncertainty estimates and learning the noise and smoothness parameters from training data. In solving RL problems through on-going interactions, the experience can be collected in data sets as $\{\mathbf{X},\mathbf{Y}\}$, where $\mathbf{X}:\{\mathbf{x}_i \in \mathbb{R}^d / i = 1,2\ldots n\}$ is the set of input vectors and $\mathbf{Y}:\{y_i \in \mathbb{R} / i = 1,2\ldots n\}$ is the set of the corresponding observations; where n is the number of samples. Assume that between pairs of input–output data there is a functional relationship $h:\mathbb{R}^d \to \mathbb{R}$, such that $y_i = h(\mathbf{x}_i) + \varepsilon, \varepsilon \sim \mathcal{N}(0,\sigma_\varepsilon^2)$. Then, in order to make inference about a function value $h(\mathbf{x}^*)$ for an unknown input $\mathbf{x}^*$, an inferred model for underlying function h is needed. Within a Bayesian framework, the inference of the underlying function h is described by the posterior probability:

$$p(h|\mathbf{X},\mathbf{Y}) = \frac{p(\mathbf{Y}|h,\mathbf{X})p(h)}{p(\mathbf{Y}|\mathbf{X})} \tag{5}$$

where $p(\mathbf{Y}|h,\mathbf{X})$ is the likelihood and $p(h)$ is a prior distribution on plausible value functions assumed by the GP model. The term $p(\mathbf{Y}|\mathbf{X})$ is called the *evidence* or *the marginal likelihood*. When modeling with GPs, a GP prior $p(h)$ is placed directly in the space of functions without the necessity to consider an explicit parameterization of the approximating function h. This prior typically reflects assumptions on the, at least locally, smoothness of h.

Similar to a Gaussian distribution, which is fully specified by a mean vector and a covariance matrix, a GP is specified by a *mean* function $m(\cdot)$ and a *covariance* function $Cov(\cdot,\cdot)$, also known as a *kernel*. A GP can be considered a distribution over functions. However, considering a function as an infinitely long vector, all necessary computations for inference and prediction of value functions can be broken down to manipulate well-known Gaussian distributions. The fact that function $h(\cdot)$ is GP distributed is indicated by $h(\cdot) \sim \mathcal{GP}_h(m, Cov)$ hereafter.

Given a GP model of the function h, we are interested in predicting the value function for an arbitrary input $\mathbf{x}_t^*$. The predictive (marginal) distribution of the function value $h_* = h(\mathbf{x}_t^*) \sim \mathcal{GP}_h(\mathbf{x}_t^*)$ for a test input $\mathbf{x}_t^*$ is Gaussian distributed with mean and variance given by:

$$\mathrm{E}\left[h(\mathbf{x}_t^*)\right] = Cov(\mathbf{x}_t^*,\mathbf{X}) + \left(\mathbf{K} + \sigma_\varepsilon^2\mathbf{I}\right)^{-1}\mathbf{Y} \tag{6}$$

$$Var\left[h(\mathbf{x}_t^*)\right] = Cov(\mathbf{x}_t^*,\mathbf{x}_t^*) + Cov(\mathbf{x}_t^*,\mathbf{X})\left(\mathbf{K} + \sigma_\varepsilon^2\mathbf{I}\right)^{-1}Cov(\mathbf{X},\mathbf{x}_t^*) \tag{7}$$

where $\mathbf{K}$ is the kernel matrix with $K_{ij} = Cov\left(\mathbf{x}_t^i,\mathbf{x}_t^j\right)\ \forall \mathbf{x}_t \in \mathbf{X}$. A common covariance function is the squared exponential (SE):

$$Cov_{SE}\left(\mathbf{x}_t^i,\mathbf{x}_t^j\right) := \zeta^2 \exp\left[-\frac{1}{2}\left(\mathbf{x}_t^i - \mathbf{x}_t^j\right)^T \mathbf{\Lambda}\left(\mathbf{x}_t^i - \mathbf{x}_t^j\right)\right] \tag{8}$$

where matrix $\mathbf{\Lambda} = diag([\ell_1^2,\ell_2^2,\ldots,\ell_d^2])$ and ℓ_r, $r = 1,\ldots,d$, being the characteristic length scales. The parameter ζ^2 describes the variability of the inductive model h. The parameters of the covariance function are the *hyperparameters* of the $\mathcal{GP}_h$ and are collected within the vector ψ. To fit parameters to value function data the evidence maximization or *marginal likelihood optimization* approach is recommended (see Rasmussen & Williams (2006)). The log-evidence is given by:

$$\begin{aligned}\log p(\mathbf{Y}|\mathbf{X},\boldsymbol{\psi}) &= \int (\mathbf{Y}|h(\mathbf{X}),\mathbf{X},\boldsymbol{\psi})p(h(\mathbf{X})|\mathbf{X},\boldsymbol{\psi})dh \\ &= -\frac{1}{2}\mathbf{Y}^T\left(\mathbf{K}_\psi + \sigma_\varepsilon^2\mathbf{I}\right)^{-1}\mathbf{Y} - \frac{1}{2}\log\left|\left(\mathbf{K}_\psi + \sigma_\varepsilon^2\mathbf{I}\right)\right| - \frac{d}{2}\log(2\pi)\end{aligned} \tag{9}$$

In Eq. (9), we have $h(\mathbf{X}) = [h(\mathbf{x}_1),\ldots,h(\mathbf{x}_n)]$ where n is the number of training points. We made the dependency of $\mathbf{K}$ on the hyper-parameters $\boldsymbol{\psi}$ explicit by writing $\mathbf{K}_{\boldsymbol{\psi}}$ in Eq. (9). Evidence maximization yields an inductive model of the value function that: (*i*) rewards data, and (*ii*) rewards simplicity of the fitted model. Hence, it automatically implements *Occam's razor*, i.e. preferring the simplest model. Further details on GP regression can be found in the excellent books of Rasmussen and Williams (2006) and MacKay (2003), and references therein.

2.2.1. Modeling the state transition dynamics

It is widely recognized that model-based methods often make better use of available information since they capture the underlying pattern (latent function) for state transitions. For controlling glycemic variability, model-based learning methods require a model of the state transition dynamics (in our case a model of the glucose–insulin dynamics) to describe essential information about the behavior of the system and the influence of an action taken on this system. A transition function f, maps a state-action pair $(\mathbf{x}_t, \mathrm{u}_t)$ to a successor glycemic state $(\mathbf{x}_{t+1})$, as is indicated in Eq. (10).

$$\mathbf{x}_{t+1} = \mathbf{x}_t + f(\mathbf{x}_t, \mathrm{u}_t) \tag{10}$$

Probabilistic models such as GPs can be used to model the glucose–insulin dynamics by learning a transition function f based on training data consisting of a sequence of observed glycemic states and changes to the insulin infusion rate (control actions). As a result, the GP learns to predict the change between two consecutive states conditioned on the previous state and the control input. Thus, the GP (inductive) model, *the dynamics*GP, is learned to describe the state transition dynamics $f \sim \mathcal{GP}_f$ through expectations on changes caused by a control action execution. We attempt to model short term transition dynamics based on the data coming from the interactions (with a real patient or a simulator). We assume that the glucose dynamics smoothly evolve while a control policy is being applied. However, we implicitly assume that glucose variability is due to uncertainty about both inter- and intra-patient variability, inaccuracies in glucose sensing and delays in insulin absorption from the subcutaneous tissue. For predicting the glucose transition dynamics, a GP model is trained in such a way the effect of uncertainty about the state transition following a control action is modeled statistically as

$$x_{t+1}^i - x_t^i \sim \mathcal{GP}_f(m_f, cov_f) \tag{11}$$

where m_f is the mean function and cov_f is the covariance function. Notice that this model implies that the output dimensions are conditionally independent on the given inputs. Moreover, the correlation between the state variables is implicitly considered when we observe pairs of states and successor states. The training inputs to the model transition dynamics $\mathcal{GP}_f$ are tuples $(\mathbf{x}_t, \mathrm{u}_t)$, whereas the targets are the state differences shown in Eq. (11). The posterior distribution for the dynamics GP reveals the remaining uncertainty about the underlying latent function for any blood glucose change caused by a control u_t when it is implemented at $\mathbf{x}_t$. GP models of the transition dynamics, for observable states, are built on the fly using data gathered from interactions with a certain patient (real or virtual).

2.2.2. GP modeling of a value function

For generalization and on-line learning in RL algorithms with continuous state and action spaces, GP models are useful to approximate the value functions $V(\cdot)$ and $Q(\cdot,\cdot)$ directly in function space by representing them by fully probabilistic GP models (Deisenroth et al., 2009). These inductive models make intuitive sense as they use data (coming from real or simulated interactions) to determine the underlying structures of the value functions, which are *a priori* unknown. Moreover, GPs provide information about confidence

intervals for the value function predictions. Thus, the state-value function $V(\cdot)$ and the action-state value function $Q(\cdot,\cdot)$ are approximated as $V(\cdot) \sim \mathcal{GP}_v(m_v, Cov_v)$ and $Q(\cdot,\cdot) \sim \mathcal{GP}_q(m_q, Cov_q)$, respectively. The training targets (observations) can be iteratively determined by the RL algorithm. Thus, the training inputs for the data-driven model $\mathcal{GP}_v$ are the states $\mathbf{x}_t$ whereas the targets are the corresponding state values $V(\mathbf{x}_t)$. Similarly, the training inputs for $G\mathcal{P}_q$ are tuples $(\mathbf{x}_t, \mathrm{u}_t)$ and the targets are the state-action values $Q(\mathbf{x}_t, \mathrm{u}_t)$. The advantage of modeling the value functions by GP models is that the GPs provides a predictive distribution of $V(\mathbf{x}_t)$ or $Q(\mathbf{x}_t, \mathrm{u}_t)$ for any $\mathbf{x}_t$ or any pair $(\mathbf{x}_t, \mathrm{u}_t)$ through Eqs. (6) and (7). Due to the generalization property of $\mathcal{GP}_v$ and $\mathcal{GP}_q$, we can work in a continuous state space $\mathbb{X} \subseteq \mathbb{R}^{n_x}$ and action space $\mathbb{U} \subseteq \mathbb{R}$ to determine a value function $V(\mathbf{x}^*)$ or $Q(\mathbf{x}^*, \mathrm{u}^*)$ for any state $\mathbf{x}^*$ or state-action pair $(\mathbf{x}^*, \mathrm{u}^*)$. Then, to obtain the optimal control from the value functions for a state $\mathbf{x}_t$ in Eq. (4), the optimal action is given by the maximum of the mean function of $\mathcal{GP}_{\Pi}$ which is obtained by solving the optimization problem:

$$\begin{aligned} &\pi^*(\mathbf{x}_t) = \mathrm{argmax}_{\mathrm{u}} Q^*(\mathbf{x}_t, \mathrm{u}) = \mathrm{argmax}_{\mathrm{u}} m_q(\mathrm{u}) \\ &\text{subject to}: \ \mathrm{u}^{\min}(\mathbf{x}_t) \leqslant \mathrm{u} \leqslant \mathrm{u}^{\max}(\mathbf{x}_t) \end{aligned} \tag{12}$$

It is worth noting that using a probabilistic model for the value function allows addressing uncertainty in state transitions following a control action and the variability in corresponding rewards (costs) in a natural way. Also, by resorting to GPs, the proposed RL algorithm (developed below) can readily handle uncertainty in state transitions due to a stochastic disturbance dynamics along with noisy measurements and rewards.

2.2.3. GP modeling of a control policy

We regard the optimal policy π^* as a probabilistic map from states to actions. In order to generalize the control policy all over the state space, we have to solve a regression problem to obtain an optimal policy π^* based on an observed data set. These data are obtained from a sequence of interactions (real or simulated) between a learning controller and a given patient. Although any function approximation technique can be used to model the control policy, here it is also approximated with a GP model, $\mathcal{GP}_\pi$, since it allows assessing the remaining uncertainty when estimating optimal controls for unseen states. This capability is instrumental in real-time personalization of a generic control policy. The training inputs of the model $\mathcal{GP}_\pi$ are states $\mathbf{x}_t \in \mathbb{X} \subseteq \mathbb{R}^{n_x}$ whereas the training targets are controls $\mathrm{u}_t \in \mathbb{U} \subseteq \mathbb{R}$. When the targets are the optimal controls $\mathrm{u}_t^* = \pi^*(\mathbf{x}_t)$ obtained from Eq. (12), the $\mathcal{GP}_\pi$ models the optimal control policy.

2.3. Generic control policy

A generic control policy is a policy which is obtained in an off-line way employing a simulation environment. Through a simulation experiment, involving the interaction between a given controller and a patient, under certain conditions, valuable training data can be acquired. This data can be sorted and grouped in a support data set $\mathbb{D}$, or simply support set, such that $\mathbb{D} = \{\mathscr{A}_i\}_{i=1}^n, \mathscr{A}_i = [\mathbf{x}_t \ \mathrm{u}_t] \ \forall \ \mathbf{x}_t \in \boldsymbol{\mathcal{X}}$ and $\forall \ \mathrm{u}_t \in \mathrm{U}$, where $\mathbb{X} \subset \mathbb{D}$ and $\mathbf{U} \subset \mathbb{D}$ are the support sets of training inputs and targets respectively. Thus, based on a certain support set, a generic policy can be learned and modeled by a GP model, $\mathcal{GP}_\pi$, in on-line off-line way. During the experiment, any feedback control strategy can be applied to generate the training data. However, to introduce our approach we employ the RL algorithm -referred to as Algorithm 1- sketched in the Fig. 1 to learn a generic control policy $\pi^* \sim \mathcal{GP}_\pi$.

In Fig. 1, an algorithm integrating RL with GP is depicted. As input, an initial policy π_0 has to be supplied. In a general situation, when no knowledge about the control task exists, π_0 is commonly a random policy which actually consists in a random function to select admissible actions. In line 7, using a simulation environment of the dynamics function f (which simulate the insulin–glucose dynamics) the current policy π_{k-1} is applied to the simulator from each start state $\mathbf{x}_0 \in \mathcal{X}_0$ during $\mathcal{T}$ sampling times, obtaining the support sets $\mathbb{X}$ and $\mathbf{U}$. For policy iteration, this algorithm describes the value functions $V(\cdot)$ and $Q(\cdot,\cdot)$ directly in function space by representing them by fully probabilistic GP models. These inductive models make intuitive sense as they use simulation data to determine the underlying structures of these value functions, which are *a priori* unknown. The sets $\boldsymbol{\mathcal{X}}$ and $\mathbf{U}$ instead of being a discrete representation for the state and action spaces, are considered the *support points* (training data) to approximate the value functions $V(\cdot)$ and $Q(\cdot,\cdot)$ using GP models; which are indicated as $V(\cdot) \sim \mathcal{GP}_v$ and $Q(\mathbf{x}_i, \cdot) \sim \mathcal{GP}_q$, respectively. The training targets (observations) are iteratively determined by Algorithm 1 itself.

As it has been already mentioned, the advantage of modeling the state-value function $V(\cdot)$ by $\mathcal{GP}_v$ is that the GP provides a predictive distribution of $V(\mathbf{x}_t)$ for *any* state $\mathbf{x}_t$. This property is exploited in the computation of the Q-value (line 12). Due to the generalization property of $\mathcal{GP}_v$ we are not restricted to a finite set of successor states $\mathbf{x}_i'$, result of applying an action u_j at a certain state $\mathbf{x}_i$, when determining $E[V(\mathbf{x}_i')|\mathbf{x}_i, \mathrm{u}_j, f, \mathcal{GP}_v]$.

Once the whole recursion for obtaining a support set $\boldsymbol{\mathcal{X}}$ and updated optimal controls $\pi^*(\boldsymbol{\mathcal{X}})$ have been completed, a new version of the control policy is obtained in the form of a Gaussian process, *the policy*GP: $\mathcal{GP}_{\pi_k}$ in line 18. Policy iteration converges when distributions for control policies are "close" enough. As control policies in successive iterations are also modeled using GPs, policy iteration can be stopped when the relation ε of the *Kullback–Leibler* (KL) divergences over a support set, with cardinality $\|\boldsymbol{\mathcal{X}}\|$, between two successive policy GPs is lower than a small tolerance δ. Typically, only a few iterations are necessary for the control policy distributions to converge according to the criterion in Eq. (13).

$$\epsilon = \frac{\mathfrak{K}_{previous} - \mathfrak{K}_{current}}{\mathfrak{K}_{previous}} \cdot 100\% \leqslant \delta \tag{13}$$

where:

$$\begin{aligned} \mathfrak{K}_{previous} &= \sum_{\mathbf{x} \in \boldsymbol{\mathcal{X}}_k} KL\left(\mathcal{GP}_{\pi^*_{k-2}(\boldsymbol{\mathcal{X}})} \mathcal{GP}_{\pi^*_{k-1}(\boldsymbol{\mathcal{X}})}\right) \Big/ \|\boldsymbol{\mathcal{X}}\| \\ &= \sum_{\mathbf{x} \in \boldsymbol{\mathcal{X}}_k} \mathcal{GP}_{\pi^*_{k-1}(\mathbf{x})} \cdot \log\left(\frac{\mathcal{GP}_{\pi^*_{k-1}(\mathbf{x})}}{\mathcal{GP}_{\pi^*_{k-2}(\mathbf{x})}}\right) \Big/ \|\boldsymbol{\mathcal{X}}\| \end{aligned} \tag{14}$$

$$\begin{aligned} \mathfrak{K}_{current} &= \sum_{\mathbf{x} \in \boldsymbol{\mathcal{X}}_k} KL\left(\mathcal{GP}_{\pi^*_{k-1}(\mathbb{X})} \mathcal{GP}_{\pi^*_p(\boldsymbol{\mathcal{X}})}\right) \Big/ \|\boldsymbol{\mathcal{X}}\| \\ &= \sum_{\mathbf{x} \in \mathbb{X}} \mathcal{GP}_{\pi^*_k(\mathbf{x})} \cdot \log\left(\frac{\mathcal{GP}_{\pi^*_k(\mathbf{x})}}{\mathcal{GP}_{\pi^*_{k-1}(\mathbf{x})}}\right) \Big/ \|\boldsymbol{\mathcal{X}}\| \end{aligned} \tag{15}$$

As it is shown in the line 21 of Algorithm 1 (Fig. 1), a GP model is learned to approximate the generic policy π^* based on the support sets $\boldsymbol{\mathcal{X}}$ and $\pi^*(\boldsymbol{\mathcal{X}})$. As it was said, these data sets can be obtained by solving a RL problem with the proposed Algorithm 1, or by recording input–output data coming from simulations of applying other control strategies. The simulation environment to emulate the glycemic behavior in a diabetic patient is generally representative of only an average subject under specific conditions. Therefore, once a generic policy is obtained, it can be safely applied to a certain patient. However, it is imperative to adapt this policy to the patient-specific behavior and lifestyle. To do this, in the next section we propose a methodology for control policy personalization to control a given patient glucose variability using simulation-based learning and on-line adaptation of a generic control

Algorithm 1. RLGP

1. Input $\mathcal{X}_0 = \{\mathbf{x}_0^l\}_{l=1}^{n_0}, \mathcal{T}, f, \pi_0, \delta, r(\cdot), \gamma$
2. $k = 0$
3. $\epsilon = \infty$
4. **Until** $\epsilon \leq \delta$ do
5. $k = k + 1$
6. $\mathfrak{X} = \phi$; $\mathbf{U} = \phi$
7. Simulate $\mathcal{T}$ transitions using f, starting from each $\mathbf{x}_0 \in \mathcal{X}_0$, applying π_{k-1}. Record the observed state transitions ($\mathbf{x}_t$) and applied actions (u_t) in $\mathfrak{X}$, $\mathbf{U}$.
8. Compute sequences of rewards $\{r_t\}_{t=1}^{t=\mathcal{T}.n_0}$ for the n_0 trajectories and estimate $V(\mathbf{x}_t)\ \forall\ \mathbf{x}_t \in \mathfrak{X}$ using Eq. (1) obtaining $\mathbf{V}_{\mathfrak{X}} = \{V(\mathbf{x}_t)\}_{t=1}^{t=\mathcal{T}.n_0}$.
9. Approximate $V(\cdot) \sim \mathcal{GP}_v$ with $\mathfrak{X}$, $\mathbf{V}_{\mathfrak{X}}$
10. **For all** $\mathbf{x}_i \in \mathfrak{X}$ **do**
11. **For all** $\mathbf{u}_j \in \mathbf{U}$
12. $Q(\mathbf{x}_i, \mathrm{u}_j) = r(\mathbf{x}_i, \mathrm{u}_j) + \gamma\, E[V(\mathbf{x}_i')|\mathbf{x}_i, \mathrm{u}_j, f, \mathcal{GP}_v]$
13. **End For**
14. $Q(\mathbf{x}_i, \cdot) \sim \mathcal{GP}_q$
15. $\pi^*(\mathbf{x}_i) \in \mathrm{argmax}_{\mathbf{u}}\ Q^*(\mathbf{x}_t, \mathbf{u}) = \arg\max_{\mathbf{u}} m_q(\mathbf{u})$
16. **End For**
17. $\pi^*(\mathfrak{X}) = \{\pi^*(\mathbf{x}_i)\}\ \forall\ \mathbf{x}_i \in \mathfrak{X}$
18. Approximate $\pi_k \sim \mathcal{GP}_{\pi_k}$ with $\mathfrak{X}$, $\pi^*(\mathfrak{X})$
19. $\epsilon = \frac{\Re_{previous} - \Re_{current}}{\Re_{previous}} \cdot 100\% \leq \delta$ Eq. (13)
20. **End Loop**
21. **Return** a generic policy $\pi^* = \pi_k \sim \mathcal{GP}_{\pi_k}$

Fig. 1. Integration of reinforcement learning with Gaussian processes for policy iteration.

policy. The proposed approach is based on reinforcement learning principles, it is integrated with a recently methodology for incremental online sparsification and employs Gaussian process approximation. Also, a Bayesian active learning mechanism is proposed for a self-exploration process (Baranes & Oudeyer, 2013) able to work in an on-line way with a continuous data stream in real-time.

It is important to highlight a key advantage of resorting to Gaussian processes for data-driven modeling of control policies. Using GPs, a generic control policy can be modeled over an augmented support set, $\mathbb{D}^*$, which combine the support sets of different generic policies obtained from different simulation runs. For example, each simulation experiment can be done under different conditions, for instance employing different control strategies, regarding different patient configurations, assuming different meal routines as well as many other possible variants. Thus, each jth simulation experiment will give rise to a different support set $\mathbb{D}^j$. Thus, an augmented support set can be obtained as: $\mathbb{D}^* = \{\mathbb{D}^1 \cup \mathbb{D}^2 \cup \ldots \cup \mathbb{D}^j \cup \ldots\}$. Then, a generic policy modeled by a GP learned over a set $\mathbb{D}^*$ should have a generalization ability to deal with many different situations that a patient may experience.

2.4. Simulation environment

To emulate diabetic patients' behavior, a simulation environment able to replicate the glucose–insulin dynamics (function f in Algorithm 1 in Fig. 1) is needed. Glucose–insulin dynamics shows great variability from patient to patient (inter-patient variability). Moreover, in the same person the glucose level evolutions are not identical along consecutive days although the patient is subjected to the same conditions every day (feeding, physical exercise, insulin infusion). This is referred to as intra-patient variability. When a control policy is learnt off-line using computational simulation, the natural inter- and intra-patient variability needs to be addressed.

Most of the glucose–insulin models proposed are based on either the Bergman's minimal model (Bergman, Ider, Bowden, & Cobelli, 1979, 1981) or Sorensen's physiological model (Sorensen, 1985). These models offer a rather qualitative prediction tool for blood glucose dynamics to account for exogenous insulin infusions and carbohydrate intake. If the uncertainties are omitted and if the model cannot accurately represent the glucose and insulin dynamics, a significant performance degradation in model-based control

strategies may arise whereas hypoglycemic episodes cannot be ruled out. Meals and exercise along with age and weight require different values of model parameters to describe in quantitative terms the variability observed in the insulin–glucose dynamics. Furthermore, patients with diabetes are constantly exposed to external disturbances that cause large blood glucose perturbations like meal consumption or physical activity on a daily basis. Both Bergman's and Sorensen's models are physiologically based and can qualitatively explain what happens with varying levels of physical activity. In this paper, the validated stochastic model proposed by Ulas Acikgoz and Diwekar (2010) is used to simulate the glucose–insulin dynamic system (simulated patients) which accounts for the different sources of uncertainties and variability. This model is an enhanced model based on the physiological model proposed by Lehmann and Deutsch (1992).

The physiological model proposed by Lehmann and Deutsch considers two subsystems (compartments) to represent the glucose–insulin dynamics based on the following system of differential equations (more detail can be found in Lehmann & Deutsch (1992) and in the website http://www.2aida.net):

$$\frac{dG}{dt} = \frac{G_{in}(t) + NHGB(p_3, I)}{V_G} - \frac{G(p_2 I_a + p_5)(K_M + G_X)}{G_X(K_M + G)V_G} - \frac{G_{ren}}{V_G} \tag{16}$$

$$\frac{dI_a}{dt} = p_3 I - p_4 I_a \tag{17}$$

$$\frac{dI}{dt} = -\eta I + \frac{D_t}{V_I} \tag{18}$$

In the differential equations system above, Eqs. (16)–(18) represent the changes with time of the: plasma glucose concentration$G$, insulin in remote compartment I_a and plasma insulin concentration I, respectively; their initial conditions are: $G(0)$,$I_a(0)$ and $I(0)$. In Eq. (16) G_{in} is the systemic appearance of glucose via glucose absorption from the gut, $NHGB$ is the net hepatic glucose balance, V_G is the volume of distribution of glucose, G_X is a reference glucose level, K_M is the Michaelis–Menten constant between glucose utilization and plasma glucose concentration and G_{ren} is the renal excretion of glucose. In Eq. (18), η is the fractional disappearance rate of insulin, V_I is the insulin distribution volume and D_t is the flow rate of exogenous insulin infused such that $D_t = D_{t-1} + u_t$ where u_t is the change in the insulin supplied (control action). The vector $P = [p_1 p_2 p_3 p_4 p_5]$ stands for estimated patient-specific parameters and in this work the values estimated by Ulas Acikgoz and Diwekar (2010) are used. These parameters are given in Table 1, assuming a body weight of 70 kg. The parameter Sh that appears in Table 1 is the hepatic sensitivity needed to determine the $NHGB(t)$ (see Lehmann & Deutsch (1992)).

Eqs. (16)–(18) provide a deterministic dynamic model for blood glucose in Type 1 diabetic patients. However, there exists uncertainty about the estimation of model parameters in Table 1 that prevents describing variability among daily values of glucose in patients, whereas other sources of structural errors give rise to model-patient mismatch. Also, in a standard implementation there exists inaccuracies of the measurement devices and therefore the available measurements do not uniquely determine the true state in a diabetic patient. Thus, there is uncertainty in both estimating system state and predicting the outcome of control actions. Furthermore, noise is present in subcutaneous glucose sensors which is another important cause of variability (Sanger, 2010). All these sources of time-dependent uncertainties could be represented by introducing stochastic processes in a deterministic model (Ulas Acikgoz & Diwekar, 2010). Introducing a superimposed Ito's stochastic process (Ito, 1951), to represent the variability in plasma glucose concentration, can be obtained by modifying Eq. (16) as follows:

$$\frac{dG}{dt} = \frac{G_{in}(t) + NHGB(p_3, I)}{V_G} - \frac{G(p_2 I_a + p_5)(K_M + G_X)}{G_X(K_M + G)V_G} - \frac{G_{ren}}{V_G} + \frac{\sigma_{\text{ito}}\varepsilon}{\sqrt{dt}};$$
$$G(0) = G_0 \tag{19}$$

where σ_{ito} is the variance parameter and ε is a random number generated by a normal distribution with a mean equal to zero and a standard deviation equal to one ($\varepsilon \sim \mathcal{N}(0,1)$).

The preceding model was used to represent the dynamics as the function f describing an average glycemic behavior of diabetic patients with parameters in Table 1. The inter- and intra-variability in diabetic patients are modeled using $\sigma_{\text{ito}} = 0.25$. Later on, the Algorithm 1 (Fig. 1) is applied and a generic control policy π^* is obtained. In Fig. 2, the key concept of a control policy π^* is highlighted through the resulting glucose profiles of 125 independent simulations made using the described dynamics model (Eqs. (19), (17) and (18) manipulated by an optimal policy π^*. It is noticeable that the optimal policy is able to achieve tight glycemic control as glucose levels are within soft constrains (green lines).

The next section presents a novel simulation-based algorithm for real-time personalization of a generic control policy (π^*) to the specific characteristics and lifestyle of a given patient.

Table 1
Model parameters.

Parameter	Value	Unit
V_G	0.22	L/kg
G_X	5.3	mmol/L
K_M	10.0	mmol/L
η	5.4	h^{-1}
V_I	0.1421	L/kg
p_1	0.278	
p_2	0.0248	mmol min^{-1} kg^{-1} mU^{-1} L^{-1}
p_3	0.000758	min^{-1}
p_4	0.0148	min^{-1}
p_5	0.00986	mmol min^{-1} kg^{-1}
Sh	0.5	

3. On-line policy learning and adaptation

Once a generic control policy (π^*) has been obtained in an off-line way, it can be adapted to the requirements of a specific patient so as to obtain a personalized control policy, referred to as π_c^* hereafter. By introducing some changes to the Algorithm 1 in Fig. 1, a new version capable of adapting a generic control policy whilst the control algorithm interacts with the patient is obtained. Bayesian active learning, borrowed from robotics, should be included for safe exploration as the control policy is being implemented (Baranes & Oudeyer, 2013; Cohn et al., 1996; Settles, 2010). Hence, only a relevant part of the state space will be explored while the exploitation–exploration dilemma of the RL problem is addressed (Krause & Guestrin, 2007). Moreover, during the on-going interactions between the AP (and its control policy) and a patient, a model of the glucose dynamics must be learnt on-line, allowing the policy to adapt itself to changes in the patient response to control actions. For this reason, the on-line learning algorithm must be capable of dealing with continuous data stream to update the support data set. To this aim, incremental on-line sparsification suitable to work with non-parametric Gaussian process regression is required (Engel et al., 2004; Nguyen-Tuong & Peters, 2011a). The incremental sparsification technique makes possible to determine whether arriving data provide valuable information regarding the current support sets for both the dynamics and the policy. This allows us to limit the cardinality of the support sets and keep them updated to account for subject-specific glycemic variability.

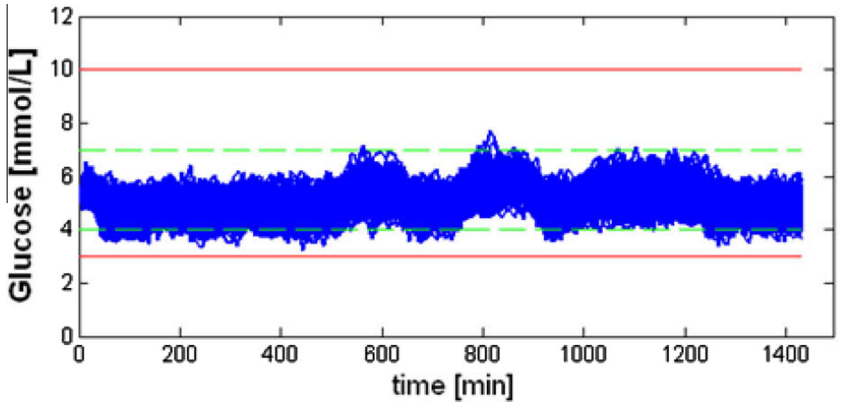

Fig. 2. Glucose profiles for 125 independent simulations under the generic policy π^*.

3.1. Bayesian active learning

To learn probabilistic metamodels of the glucose–insulin dynamics and the value functions on the fly, a constrained version of Bayesian active learning (BAL) mechanism must be used on-line. Let us first describe how to determine the most promising states from a given set of candidate states ($\widetilde{\mathcal{X}}$) and then a mechanism suitable for online interactions with the patient (real or virtual) is proposed to obtain candidate data to be added to the support sets. A version of Bayesian active learning algorithm is sketched in Fig. 3 which is referred to as BAL algorithm or Algorithm 2.

3.1.1. Selecting promising states

Consider a given set $\widetilde{\mathcal{X}}$ of candidate states $\tilde{\mathbf{x}}_i, i = 1 \ldots n$. We need to value them in some way. Thus, we have to rank them according to that value and select the one with the highest value. In on-line policy learning control, a "good" state $\tilde{\mathbf{x}} \in \widetilde{\mathcal{X}}$ should provide both information gain about the latent value function and be able to reduce uncertainty about the optimal controls. Hence, an utility function $\mathcal{U}(\cdot)$ that captures both objectives to rate the quality of candidate states is used (Deisenroth et al., 2009). Accordingly, the most promising state $\tilde{\mathbf{x}}^* \in \widetilde{\mathcal{X}}$ is the one that maximizes the expected utility function defined as follows

$$\mathcal{U}(\tilde{\mathbf{x}}) = \rho E_v(V^*(\tilde{\mathbf{x}})|\mathbb{X}) + \frac{\beta}{2}\log(var_v(V^*(\tilde{\mathbf{x}})|\mathbb{X})) \tag{20}$$

where ρ and β are used to control the exploration–exploitation tradeoff, and the predictive mean and variance of the value function are

$$E_v(V^*(\tilde{\mathbf{x}})|\boldsymbol{\mathcal{X}}) = \boldsymbol{k}_v^T \boldsymbol{K}_v^{-1} V^*(\boldsymbol{\mathcal{X}}) \tag{21}$$

$$var_v(V^*(\tilde{\mathbf{x}})|\boldsymbol{\mathcal{X}}) = \boldsymbol{k}_v - \boldsymbol{k}_v^T \boldsymbol{K}_v^{-1} \boldsymbol{k}_v \tag{22}$$

where $\boldsymbol{K}_v$ is the augmented kernel matrix when including $\tilde{\mathbf{x}}$

$$\boldsymbol{K}_v = \begin{bmatrix} \boldsymbol{K}_v & \boldsymbol{k}_v \\ \boldsymbol{k}_v^T & \boldsymbol{k}_v \end{bmatrix} \tag{23}$$

and $\boldsymbol{k}_v$ is called *kernel* vector such that $\boldsymbol{k}_v = \mathrm{k}(\boldsymbol{\mathcal{X}}, \tilde{\mathbf{x}})$. Similarly, $\boldsymbol{K}_v$ is the *kernel* matrix such that $K_v = \mathrm{k}(\mathbb{X}, \mathbb{X})$ and k_v is the *kernel* value computed as $k_v = \mathrm{k}(\tilde{\mathbf{x}}, \tilde{\mathbf{x}})$. Hereafter $\mathrm{k}(\cdot,\cdot)$ is a *kernel* function as previously indicated in Eq. (8), unless otherwise stated.

The utility of a candidate state $\tilde{\mathbf{x}}$ expresses how much total reward is expected from it when acting optimally (first term in Eq. (20)) and how surprising $V^*(\tilde{\mathbf{x}})$ is expected to be given the current training inputs $\mathbb{X}$ of the GP model for V (second term). The second term in Eq. (20) is somewhat related to the expected Shannon information (entropy) of the predictive distribution $V^*(\tilde{\mathbf{x}}^*)$ or the *Kullback–Leibler* divergence (Deisenroth, 2010; Verdinelli & Kadane, 1992) between the predictive Gaussian distribution of $V^*(\tilde{\mathbf{x}})|\boldsymbol{\mathcal{X}}$ and $V^*(\boldsymbol{\mathcal{X}})$. The parameters ρ and β are used to trade off optimality in action selection with information gain which is needed for on-line policy adaptation. A large (positive) value of ρ introduces data bias towards optimal controls, whereas a large value (positive) β favors gaining information based on the predicted variance of the value function for unseen states at different decision stages.

3.1.2. Generating the promising states

Adapting the policy to the patient's dynamics requires incrementally improving the model $\mathcal{GP}_f$ of the glucose–insulin dynamics and consequently the value function model, $\mathcal{GP}_v$, will also be improved. Therefore, the support set must be updated with actually reachable states. In the proposed BAL on-line algorithm of Fig. 3, input locations are added by interacting directly with the patient. In addition, through BAL safe exploration with control actions is done whereas taking actions that may lead the patient to hazardous states are avoided. To begin with, a safe enough generic control policy (π^*) is applied. Then, as the personalization of the policy progresses, the resulting policy (π_c^*) is increasingly tailored to the patient.

Initially, the patient is in state $\mathbf{x}_0$. In each recursion of the personalization algorithm (described below), a new datum $\measuredangle$ defined as a state-action tuple ($\mathbf{x}$,u), resulting from the interaction with the patient (real or virtual) is obtained. When the patient is in a state $\mathbf{x}'$, safe exploration is done by taking actions over a certain set $\tilde{\mathbf{U}}$ that represents the remaining uncertainty about the optimal action. This set can be defined as a uniform discretization in the feasible interval for choosing an action. We propose to establish this set based on the confidence interval of two standard deviations ($2\sigma_\pi$) around the mean of the action distribution ($\bar{\mathrm{u}}$). Thus, we obtain an interval with a confidence level of 95% for optimal action selection. The mean value $\bar{\mathrm{u}}$ is given by the mean function of the current $\mathcal{GP}_\pi$ evaluated at $\mathbf{x}'$ and the standard deviation is given by the covariance function of the $\mathcal{GP}_\pi$. Lines 4 through 7 in the BAL algorithm (Fig. 3) are the steps required to define $\tilde{\mathbf{U}}$. From line 8 to 11, a procedure to obtain the set $\widetilde{\mathcal{X}}$ is described, where several one-step transitions from $\mathbf{x}'$ for all $\tilde{\mathrm{u}}_j \in \tilde{\mathbf{U}}$ are simulated. Then, by the procedure detailed in the Section 3.1.1, the most promising state $\tilde{\mathbf{x}}^* \in \widetilde{\mathcal{X}}$ (line 12) must be determined. The action $\mathrm{u}^* = \tilde{\mathrm{u}}^* \in \tilde{\mathbf{U}}$, which gives rise to the simulated transition from $\mathbf{x}'$ to $\tilde{\mathbf{x}}^*$, is considered as the best action to be applied to the patient (line 13). Once the interaction controller-patient takes place (line 14) by applying u^*, the state transition is observed, and a the new candidate datum $\measuredangle$ is defined (line 15 and 18).

Algorithm 2. BAL

1. **Input** $\mathbf{x}_0$, $\mathcal{GP}_f$, $\mathcal{GP}_\pi$, u_{min} , u_{max}
2. $\mathbf{x}' = \mathbf{x}_0$
3. Define $\widetilde{\mathfrak{X}}$ as an empty set
4. Obtain $(\bar{\mathrm{u}}, \sigma_\pi^2) \leftarrow \mathcal{GP}_\pi(\mathbf{x}')$; where $\bar{\mathrm{u}} = m_\pi(\mathbf{x}')$, $\sigma_\pi^2 = Cov(\mathbf{x}', \mathbf{x}')$
5. **If** $\bar{\mathrm{u}} - 2\sigma_\pi \geq \mathrm{u}_{min}$; $\tilde{\mathrm{u}}_{min} = \bar{\mathrm{u}} - 2\sigma_\pi$; **else** $\tilde{\mathrm{u}}_{min} = \mathrm{u}_{min}$; **End If**
6. **If** $\bar{\mathrm{u}} + 2\sigma_\pi \leq \mathrm{u}_{max}$; $\tilde{\mathrm{u}}_{max} = \bar{\mathrm{u}} + 2\sigma_\pi$; **else** $\tilde{\mathrm{u}}_{max} = \mathrm{u}_{max}$; **End If**
7. Determine $\widetilde{\mathbf{U}} = \{\tilde{\mathrm{u}}_{min}, \ldots, \tilde{\mathrm{u}}_{max}\}$
8. **For all** $\mathrm{u}_j \in \widetilde{\mathbf{U}}$ **do**
9. $\tilde{\mathbf{x}}_j \leftarrow \mathcal{GP}_f(\mathbf{x}', \tilde{\mathrm{u}}_j)$; where $\tilde{\mathbf{x}}_j = m_f(\mathbf{x}', \tilde{\mathrm{u}}_j)$
10. $\widetilde{\mathfrak{X}} = \{\widetilde{\mathfrak{X}} \cup \tilde{\mathbf{x}}_j\}$
11. **End For**
12. Determine $\tilde{\mathbf{x}}^* = \max_{\tilde{\mathbf{x}}} \mathcal{U}(\tilde{\mathbf{x}}) \quad \forall\ \tilde{\mathbf{x}} \in \widetilde{\mathfrak{X}}$
13. Obtain $\mathrm{u}^* = \tilde{\mathrm{u}}^*$as the action which gave rise to the simulated transition from $\mathbf{x}'$ to $\tilde{\mathbf{x}}^*$
14. Interact with the patient (who is in the state $\mathbf{x}'$) applying u^*
15. Observe the transition $(\mathbf{x}', \mathrm{u}^*) \rightarrow \mathbf{x}''$
16. Define a tuple $\boldsymbol{d}$ with the observed transition in the previous line
17. $\mathbf{x}' = \mathbf{x}''$
18. **Return** $\boldsymbol{d}$, $\mathbf{x}_0 = \mathbf{x}'$

Fig. 3. Bayesian active learning algorithm.

3.2. Online sparsification in a nutshell

Sparsification techniques have been successfully employed in robotics for online identification of data-driven models. Additionally, sparsification techniques have been combined with regression techniques (Sigaud, Salaün, & Padois, 2011) in order to limit the computational complexity. Particularly, in the work of Nguyen-Tuong and Peters (2011a) an incremental online sparsification (IOS) method has been proposed which is mainly inspired in the Kernel Recursive Least-Squares algorithm proposed by Engel et al. (2004). To prevent expensive computational costs in on-line policy learning, once a new datum ($\mathscr{d}_{new}$) carrying novel information is detected, the point that brings the least information from the support set ($\mathbb{D}$) should be removed. By incorporating incremental online sparsification into the approach employed by the Algorithm 1, an incremental selection of samples that are used to update the support set is possible. Accordingly, the GP models of the dynamics ($\mathcal{GP}_f$) and the value function ($\mathcal{GP}_v$) are kept updated by re-estimating their hyper-parameters. On-line learning of the models used to approximate $\mathcal{GP}_f$ and $\mathcal{GP}_v$ are critical for continuous policy adaptation to a given patient characteristics. Following, an overview about the incremental online sparsification technique is given. Further details on IOS can be found in the works of Nguyen-Tuong and Peters (2011a) and Engel et al. (2004), and references therein. Also, the reader is referred to work of Chen et al. (2013) for a novel integration of on-line sparsification with temporal difference learning.

The general notion is that the kernel matrix K_v should be fully ranked. That is to ensure that any sample in the support set cannot be linearly represented by the other samples $\mathbb{D}$. Then, for online sparsification, the basic idea is that if an arriving datum can be linearly represented by the data in the support set, it should not be added. Assuming that at time t, the support set has m data points such that $\mathbb{D} = \{\mathscr{d}_i\}_{i=1}^m$, where, by construction, $\{\phi(\mathscr{d}_i)\}_{i=1}^m$ are linearly independent feature vectors. When a new datum, $\mathscr{d}_{new}$, arrives, the key crucial to determine to decide whether it can be linearly represented by the existing data in $\mathbb{D}$. So, testing whether $\phi(\mathscr{d}_{new})$ is linearly dependent on data points is required. The linear dependence (Schölkopf et al., 1999; Schölkopf & Smola, 2001) can be measured by

$$\delta \triangleq \left\| \sum_{i=1}^{m} \mathrm{a}_i \phi(\mathscr{d}_i) - \phi(\mathscr{d}_{new}) \right\|^2 \tag{24}$$

Therefore, when $\mathscr{d}_{new}$ brings some novelty since δ exceedes a threshold η, it must be added to $\mathbb{D}$. Accordingly, this procedure aims to cover only the relevant region of state space with a limited support set. In Eq. (24) we have a vector $a = (\mathrm{a}_1, \ldots, \mathrm{a}_m)^{\mathrm{T}}$ of coefficients a_i of linear dependence which can be determined by minimizing δ as follows

$$a = \min_a [a^T K a - 2a^T k + k] \tag{25}$$

where $\boldsymbol{K} = \mathrm{k}(\mathbb{D}, \mathbb{D}), k = \mathrm{k}(\mathbb{D}, \mathscr{d}_{new})$ and $k = \mathrm{k}(\mathscr{d}_{new}, \mathscr{d}_{new})$. Solving the unconstrained optimization problem of Eq. (25) yields the optimal $\boldsymbol{a} = \boldsymbol{K}^{-1}\boldsymbol{k}$. Replacing this result in Eq. (24) gives rise to

$$\delta = \mathrm{k}(\mathscr{d}_{new}, \mathscr{d}_{new}) - \boldsymbol{k}^T\boldsymbol{a} \tag{26}$$

Once δ is computed by Eq. (26) for a new data point, its value is compared with the threshold η and a decision to incorporate $\mathscr{d}_{new}$ into the set $\mathbb{D}$ taken. In the pseudocode of Algorithm 3 in Fig. 4 are the main steps for the sparsification procedure.

In the sparsification algorithm, when $\delta > \eta$ and the support set size is smaller than its maximum size (N_{max}), $\mathbb{D}$ must be updated using the Algorithm 3.1 in Fig. 5. Whenever the number of data in the set $\mathbb{D}$ is N_{max}, $\mathbb{D}$ must be updated by replacing an old point $\mathscr{d}_j \in \mathbb{D}$ by the new one $\mathscr{d}_{new}$. In the latter case, the Algorithm 3.2 of Fig. 6 which takes into account the spatial and temporal allocation of the data through a forgetting rate $\lambda^i \in [0,1]$ (Nguyen-Tuong & Peters, 2011a) is used. The forgetting rate defined as

$$\lambda^i(t) = \exp\left(-\frac{(t-t_i)^2}{2h}\right) \tag{27}$$

where t_i is the time when the point i was included into $\mathbb{D}$ and the parameter h controls the trade-off between spatial and temporal coverage. The detailed computations involved in Algorithms 3.1 and 3.2 are given in the Appendices A and B, respectively.

3.3. Algorithm for real time policy personalization

Based on the RL framework, the Algorithm 4 outlined in Fig. 7 for policy learning and adaptation is proposed. This algorithm is able to adapt on-line a policy by interacting with a patient in real time. The personalization process means passing from a generic control policy π^* to a particular personalized control policy π_c^* and keeping it updated over time. Initially, it is assumed that a generic control policy π^* which can be obtained as was indicated in Section 2.3 is available. This policy is modeled by a GP model, i.e. $\pi^* \sim \mathcal{GP}_{\pi^*}$. Moreover, the support set $\mathbb{D}$ which supports the model $\mathcal{GP}_{\pi^*}$ is known. As indicated in line 3, the generic policy π^* is applied to the patient during $\mathcal{T}$ sampling times. In this way, $\mathcal{T}$ state-action transitions are observed and recorded in a set $\mathbb{D}_\Gamma$. Next, through the sparsification algorithm, novel data points $\mathbb{D}_\Gamma$ are identified using Algorithm 3 which provides meaningful information to update $\mathbb{D}$ (line 5). Then, the models $\mathcal{GP}_f$ and $\mathcal{GP}_v$ are updated by re-estimating their corresponding hyper-parameters using the support set $\mathbb{D}$ (lines 7 to 9).

Algorithm 3. Sparsification

1: **Input**: $\mathfrak{D}$, $\boldsymbol{d}_{new}$, η, N_{max}
2: Compute $\boldsymbol{k} = \mathrm{k}(\mathfrak{D}, \boldsymbol{d}_{new})$, $\boldsymbol{a} = \boldsymbol{K}^{-1}\boldsymbol{k}$
3: Compute $\delta = \mathrm{k}(\boldsymbol{d}_{new}, \boldsymbol{d}_{new}) - \boldsymbol{k}^T\boldsymbol{a}$
4: **If** $\delta > \eta$ **then**
5: **If** $\|\mathfrak{D}\| < N_{max}$ **then**
6: update $\mathfrak{D}$ using ***Algorithm 3.1*** (Fig. 5)
7: **else**
8: update $\mathfrak{D}$ using ***Algorithm 3.2*** (Fig. 6)
9: **End If**
10: **End If**

Fig. 4. Sparsification algorithm.

Algorithm 3.1. Augmenting the support set

1: **Input**: $\mathfrak{D}$, $\boldsymbol{d}_{new}$
2: Update $\mathfrak{D} = \{\boldsymbol{d}_i\}_{i=1}^{m+1}$
3: **For** $i = 1$ **to** m
4: $\boldsymbol{k}_{m+1} = \mathrm{k}(\mathfrak{D}^i, \boldsymbol{d}_{new})$
5: $k_{m+1} = \mathrm{k}(\boldsymbol{d}_{m+1}, \boldsymbol{d}_{new})$
6: $k_{i,m+1} = \mathrm{k}(\boldsymbol{d}_i, \boldsymbol{d}_{new})$
7: $\boldsymbol{\alpha}_i = (\boldsymbol{K}_{old}^i)^{-1}\boldsymbol{k}_{m+1}$
8: $\gamma_i = k_{m+1} - \boldsymbol{k}_{m+1}^T\boldsymbol{\alpha}_i$
9: Update δ^i
10: Update $(\boldsymbol{K}_{new}^i)^{-1}$
11: **End For**

Fig. 5. Algorithm to augment the support set by incorporating a new data point.

Algorithm 3.2. Updating the support set

1: **Input**: $\mathfrak{D}$, $\boldsymbol{d}_{new}$
2: Compute $\lambda^i \; \forall \; \boldsymbol{d}_i \in \mathfrak{D}$
3: Compute $j = \min_i(\lambda^i\delta^i)$
4: Update $\mathfrak{D}$ overwriting $\boldsymbol{d}_j = \boldsymbol{d}_{new}$
5: **For** $i = 1$ **to** m
6: $\boldsymbol{k}_{m+1} = \mathrm{k}(\mathfrak{D}^i, \boldsymbol{d}_{new})$
7: $k_{m+1} = \mathrm{k}(\boldsymbol{d}_{m+1}, \boldsymbol{d}_{new})$
8: $k_{i,m+1} = \mathrm{k}(\boldsymbol{d}_i, \boldsymbol{d}_{new})$
9: $\boldsymbol{r} = \boldsymbol{k}_{m+1} - \mathrm{row}_j[\boldsymbol{K}_{old}^i]^T$
10: Update δ^i
11: Update $(\boldsymbol{K}_{new}^i)^{-1}$
12: **End For**

Fig. 6. Algorithm to update the support replacing an old data point by a new one.

On-line policy learning in Algorithm 4 takes place in the loop defined from lines 11 to 27. Through Bayesian active learning (Algorithm 2), safe exploration while interacting with the patient is made by choosing the best action for the current patient state. Once the control action is applied, the data point for observed state transition is available and recorded in $\mathscr{d}_{new}$ (line 12). By means of Algorithm 3, the support set $\mathbb{D}$ is updated iff $\mathscr{d}_{new}$ provides valuable information (line 13). Later on, the dynamics model $\mathcal{GP}_f$ is updated based on the latter version of $\mathbb{D}$ (line 14). In this way, an updated version of the model used to describe the patient response to control actions is available. Also, an improved version of the control policy adapted to the patient's glycemic variability is obtained.

In line 16, the Q-learning rule is implemented to update the Q-values regarding an immediate reward $r(\mathbf{x}_i,\mathrm{u}_i)$ and a discount factor γ. The value function and the dynamics are approximated

Algorithm 4. On-line policy learning and adaptation

1: **Input**: $\pi^* \sim \mathcal{GP}_{\pi^*}$, $\mathfrak{D} = \{\mathfrak{X}, \pi^*(\mathfrak{X})\}$, η, h, N_{max} , ρ, β, γ

2: $\pi^*_{c_0} = \pi^*$

3: Interact with the patient during $\mathcal{T}$ samples applying π^* to the patient $\rightarrow \mathbf{x}_0 = \mathbf{x}_{\mathcal{T}}$, $\mathfrak{D}_{\mathcal{T}}$

4: **For all** $\boldsymbol{d}_i \in \mathfrak{D}_{\mathcal{T}}$

5: $\mathfrak{D} \leftarrow$ ***Algorithm 3*** $(\mathfrak{D}, \boldsymbol{d}_i, \eta, h, N_{max})$ $\Rightarrow$ *Sparsification*

6: **End For**

7: Train $\mathcal{GP}_f$ with $\mathfrak{D}$

8: $V_0(\mathfrak{X}) = r(\mathfrak{X}, \pi^*(\mathfrak{X}))$ such that $\mathfrak{X} \subset \mathfrak{D}$, $\pi^*(\mathfrak{X}) \subset \mathfrak{D}$

9: Train $\mathcal{GP}_v$ with $V_0(\mathfrak{X})$ such that $\mathfrak{X} \subset \mathfrak{D}$

10: $p = 1$

11: **Loop**

12: $\boldsymbol{d}_{new} \leftarrow$ ***Algorithm 2*** $(\mathbf{x}_0, \rho, \beta, \pi^*_{c_{p-1}})$ $\Rightarrow$ *Bayesian Active Learning*

13: $\mathfrak{D} \leftarrow$ ***Algorithm 3*** $(\mathfrak{D}, \boldsymbol{d}_{new}, \eta, h, N_{max})$ $\Rightarrow$ *Sparsification*

14: Update $\mathcal{GP}_f$ with $\mathfrak{D}$

15: **For all** $(\mathbf{x}_i, \mathrm{u}_i)$ such that $\mathbf{x}_i \in \mathfrak{X} \subset \mathfrak{D}$, $\mathrm{u}_i \in \pi^*(\mathfrak{X}) \subset \mathfrak{D}$; $i = 1, \ldots, \|\mathfrak{D}\|$ **do**

16: $Q(\mathbf{x}_i, \mathrm{u}_i) = r(\mathbf{x}_i, \mathrm{u}_i) + \gamma\, \mathrm{E}[V(\mathbf{x}_i'') \mid \mathbf{x}_i, \mathrm{u}_i, \mathcal{GP}_f]$

17: **End For**

18: **For all** $\mathbf{x}_i$ such that $\mathbf{x}_i \in \mathfrak{X} \subset \mathfrak{D}$

19: $Q(\mathbf{x}_i, \cdot) \sim \mathcal{GP}_q$

20: $\pi^*(\mathbf{x}_i) \in \arg\max_{\mathrm{u}} Q(\mathbf{x}_i, \mathrm{u}) \approx \max_{\mathrm{u}} m_q(\mathrm{u})$

21: $V^*(\mathbf{x}_i) = Q\left(\mathbf{x}_i, \pi^*(\mathbf{x}_i)\right)$

22: **End For**

23 : Determine $\mathcal{GP}_v$ with $V^*(\mathfrak{X})$ such that $\mathfrak{X} \subset \mathfrak{D}$

24: Determine $\mathcal{GP}_{\pi^*_{c_p}}$ with all $\mathbf{x}_i \in \mathfrak{X} \subset \mathfrak{D}$ and $\pi^*(\mathfrak{X})$

25: $\pi^*_{c_p} := \mathcal{GP}_{\pi^*_{c_p}}$ $\Rightarrow$ *Improved control policy*

26: $p = p + 1$

27: **End Loop**

Fig. 7. On-line policy learning and adaptation algorithm.

by GP models, therefore the uncertainties about value function and dynamics have been taken into account. Due to the generalization property of $\mathcal{GP}_f$ and $\mathcal{GP}_v$, when the expected value function of $\mathbf{x}_i''$ $\left(E\left[V(\mathbf{x}_i'')|\mathbf{x}_i, \mathrm{u}_i, \mathcal{GP}_f\right]\right)$ is computed there is no need to restrict to a finite set of successor states $\mathbf{x}_i''$. Thus, in the computation of these expected values, the expected value of V^* at a successor (uncertain) state $\mathbf{x}_i''$ is estimated as described in the Appendix C (Deisenroth et al., 2009). Later on, in the next *for loop* (line 18 to 22), with the computed Q-values and all actions $\mathrm{u}_i \in \pi^*(\mathbb{X}) \subset \mathbb{D}$, a model of state-action value function, $\mathcal{GP}_q$, is learned. Then, the optimal control $\mathrm{u}_i^* = \pi^*(\mathbf{x}_i)$ is the maximizing argument of the Q-values for a certain state $\mathbf{x}_i$, and the value function $V^*(\mathbf{x}_i)$ at $\mathbf{x}_i$ is the corresponding maximum value. To maximize Q for a certain state $\mathbf{x}_i$, standard optimization methods such as Golden section or Fibonacci Search are used. Note that $\mathcal{GP}_q$ models a function of u only since $\mathbf{x}_i$ is fixed. In this way, the optimal action is obtained through $max_{\mathrm{u}} Q(\mathbf{x}_i, \mathrm{u}) \approx max_{\mathrm{u}} m_q(\mathrm{u})$, the maximum of the mean function $m_q(\mathrm{u})$ of $\mathcal{GP}_q$.

Once the optimal values $V^*(\mathbf{x}_i) \forall \mathbf{x}_i \in \mathbb{X} \subset \mathbb{D}$ are computed, a GP model, $\mathcal{GP}_v$, to approximate the state-value function is learned. Moreover, with all $\mathbf{x}_i \in \mathbb{X} \subset \mathbb{D}$ and the optimal controls computed in line 20 for all $\mathbf{x}_i \in \mathbb{X}$ an updated version of the control policy can be approximated by a new GP model, which is referred as $\mathcal{GP}_{\pi^*_{c_p}}$ in the recursion p of the Algorithm 4.

As iterations of Algorithm 4 are made, the personalization process of the control policy takes place. We say that there is a "personalization" since the dynamics GP model, $\mathcal{GP}_f$, is learned with data coming from direct interactions with the patient (real or virtual). Any change in patient's dynamic behavior will be reflected in the underlying structure of the collected data. Therefore, the dynamic model must be updated in each recursion of Algorithm 4. Hence, the flexible features of nonparametric approx-

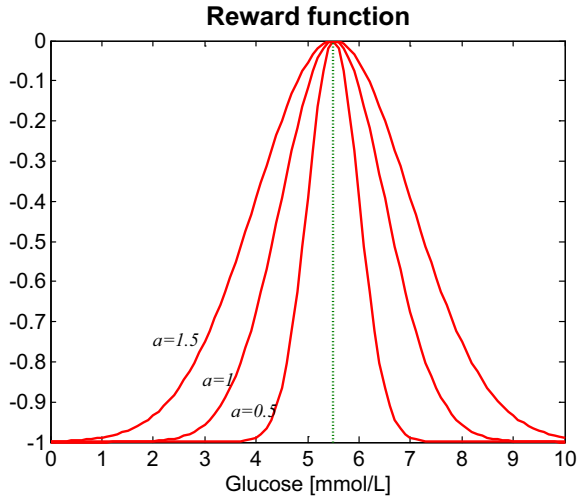

Fig. 8. Examples of Gaussian reward functions for different a parameters with $G_x = 5.5$.

imators plays a central issue. Modeling glycemic response by $\mathcal{GP}_f$ allows a nonparametric representation which can be improved continuously to the patient behavior based on data from on-going interactions. Accordingly, a policy personalization is made since the state-value function depends on the dynamic model accuracy and the policy model, $\mathcal{GP}_{\pi_c^*}$. Note that $\mathcal{GP}_{\pi_c^*}$ is learnt based on optimal controls $(\pi^*(\mathbb{X}))$, which depends on the $\mathcal{GP}_q$ model which, in turn, depends on state-actions values (computed in line 16) which are estimated using the dynamics model $\mathcal{GP}_f$.

Table 2
Feeding schedules.

	Meal routine I	Meal routine II
Meal times (min)	[180 780 960 1080]	[180 300 450 660 870 1020]
Carbohydrate content (g)	[20 20 60 20]	[47 16 63 31 63 31]

4. Simulation results

4.1. RL settings for glucose regulation

Controlling glycemia in diabetic patients means maintaining the glucose level within the normoglycemic range. In the diabetes research community there is not universal agreement about a unique and standard range for normoglycemia. For instance, in some research papers a hypoglycemic event is defined using a lower threshold of 3.33 mmol/L (∼60 mg/dl) for the glucose level, while others set the lower bound for normoglycemia at 2.22 mmol/L (∼40 mg/dl). Likewise, in the existing literature is possible to find that hyperglycemic events are defined using from 10 mmol/L (∼180 mg/dl), 15 mmol/L (∼270 mg/dl), 16.66 mmol/L (∼ 300 mg/dl) and up to 18 mmol/L (∼325 mg/dl), which is a very dangerous level. Narrow ranges of glycemic variability prevent multiple long-term complications of diabetes, e.g. the oxidative stress, that contribute to increase morbidity and mortality

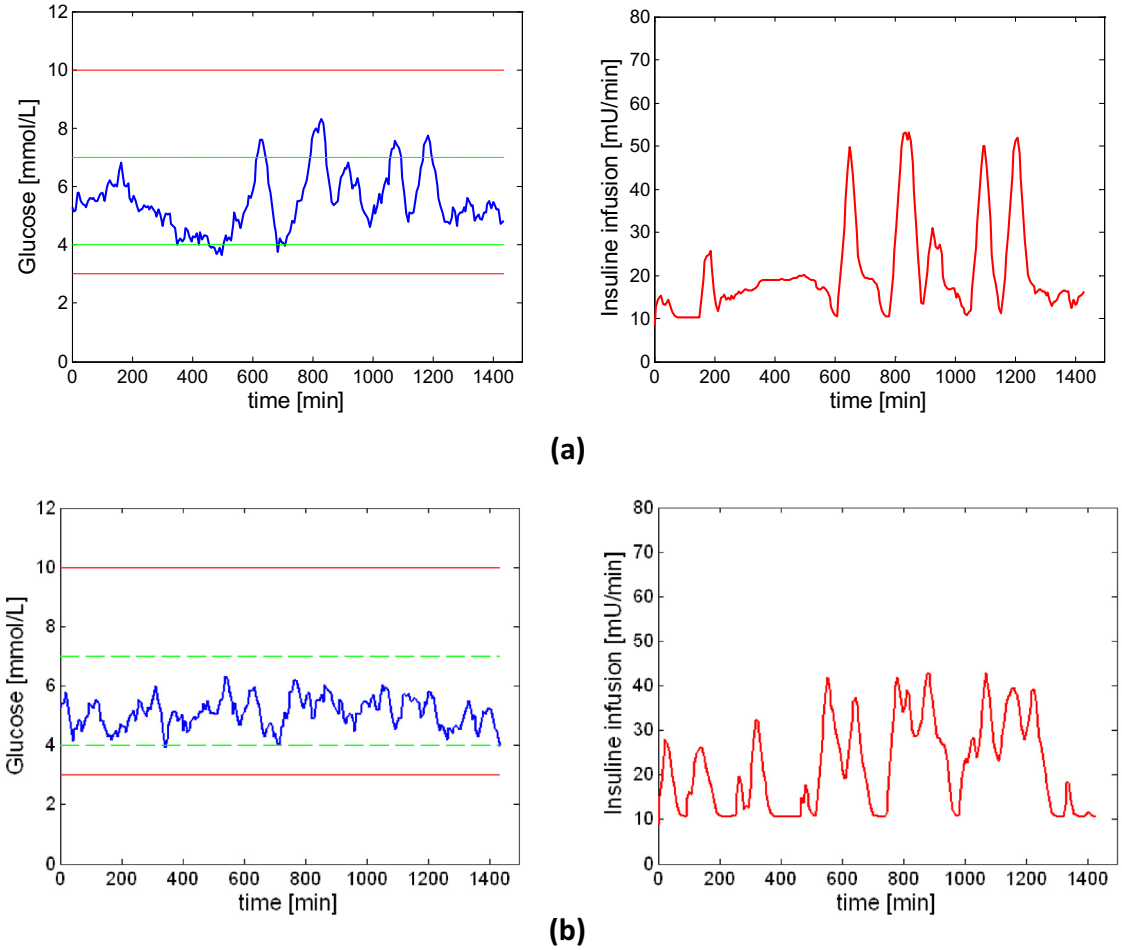

Fig. 9. Results obtained through Algorithm 1 for two different settings of the parameter a for the gaussian reward function. (a) $a = 1.5$; (b) $a = 1$.

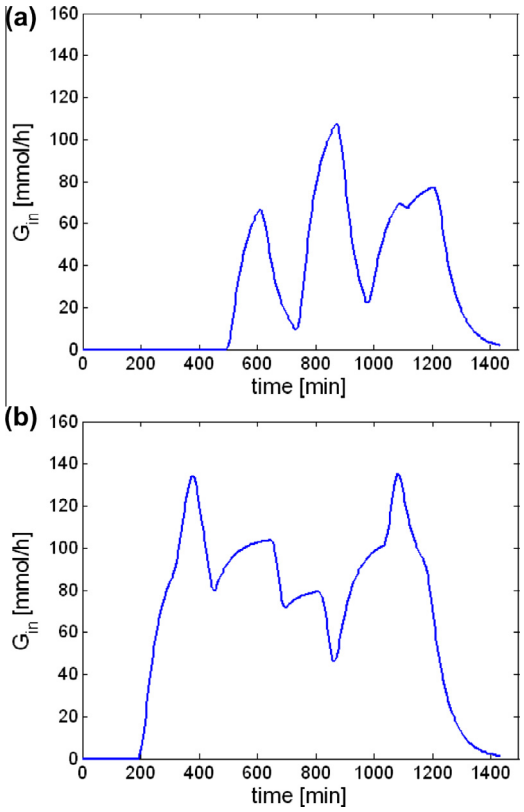

Fig. 10. Rates of glucose absorption from the gut for the feeding schedules in Table 2. (a) G_{in} for meal routine I; (b) G_{in} for meal routine II.

(Marling, Shubrook, Vernier, Wiley, & Schwartz, 2011; Siegelaar et al., 2010). For on policy learning, the normoglycemic range is defined by glucose levels between 4 mmol/L to 7 mmol/L. These tighter bounds give rise to a more difficult control task which highlights the need of addressing the long-term effects of control actions. In the RL problem formulation, the lower and upper bounds for glucose levels are set by choosing an appropriate reward function. In our simulation experiments, the following Gaussian reward function is used:

$$r(G(t)) = -1 + e^{-\frac{1}{2}\frac{(G(t)-G_x)^2}{a^2}}; \quad r \in [-1, 0] \tag{28}$$

where $G(t)$ is the instantaneous reading from the glucose sensor whereas G_x and a are the symmetry center and the amplitude of the Gaussian function $r(\cdot)$, respectively. Thereby, the reward function represented in Eq. (28) saturates for significant deviations from G_x. The width of the glucose band is defined by the parameter a. In order to enforce glucose levels between 4 mmol/L and 7 mmol/L, $G_x = 5.5$ and $a = 1$ are selected. It is worth mentioning that these limits are not *mandatory* thresholds since they may be exceeded by a small amount during short periods of time. Note that the optimal policy should generate a sequence of control actions such that the cumulative reward is maximized instead of maximizing each immediate reward. In Fig. 8, examples of different reward functions for different values of the parameter a are shown. As can be seen, as the value of a decreases, the reward function spans a smaller interval of glucose readings and consequently the control task is more challenging.

To illustrate the effect of changing the parameter a in the reward function, Fig. 9 depicts glucoses profiles resulting of applying the optimal policies found by using Algorithm 1 (in Fig. 1) for the same patient (simulated) under identical conditions (diet, exercise, etc.). For $a = 1.5$, the control policy is acceptable (see Fig. 9(a)) although rather looser when compared to the much tighter policy resulting from using $a = 1$ in the reward function (see Fig. 9(b)). As can be expected, the corresponding control policy is a bit more aggressive in the latter case to guarantee optimal performance. It is worth noting that green dotted lines in Fig. 9 correspond to *soft* thresholds for the blood glucose concentrations in the range from 4 to 7 mmol/L whereas red solid lines correspond to *hard* thresholds imposed to penalize hypo- or hyper-glycemic bounds. Hereafter, $G_x = 5.5$ and $a = 1$ will be chosen unless otherwise stated.

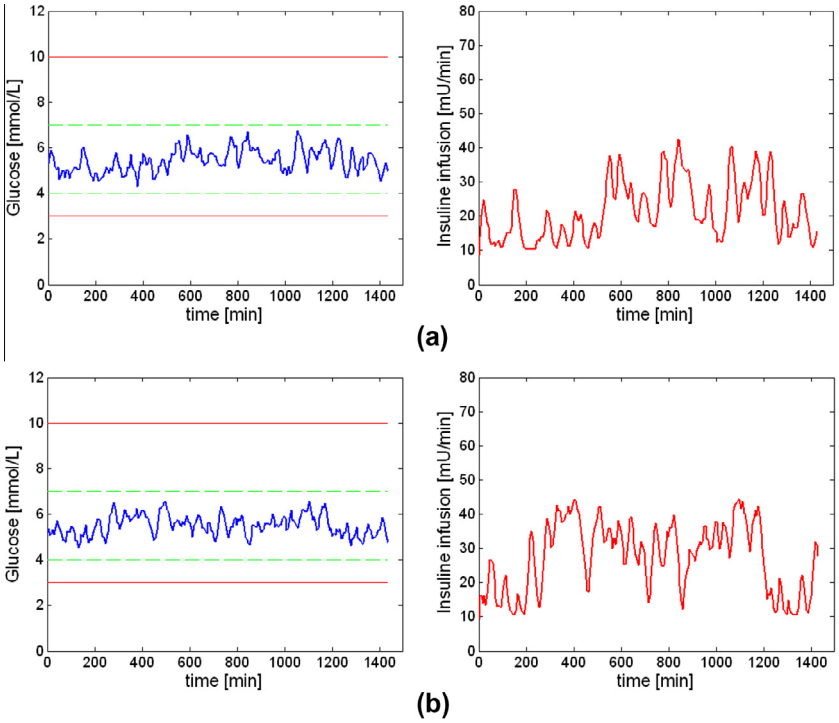

Fig. 11. Glucose profiles by applying the optimal generic policy found by Algorithm 1 on a patient (virtual). (a) Results for meal routine I; (b) Results for meal routine II.

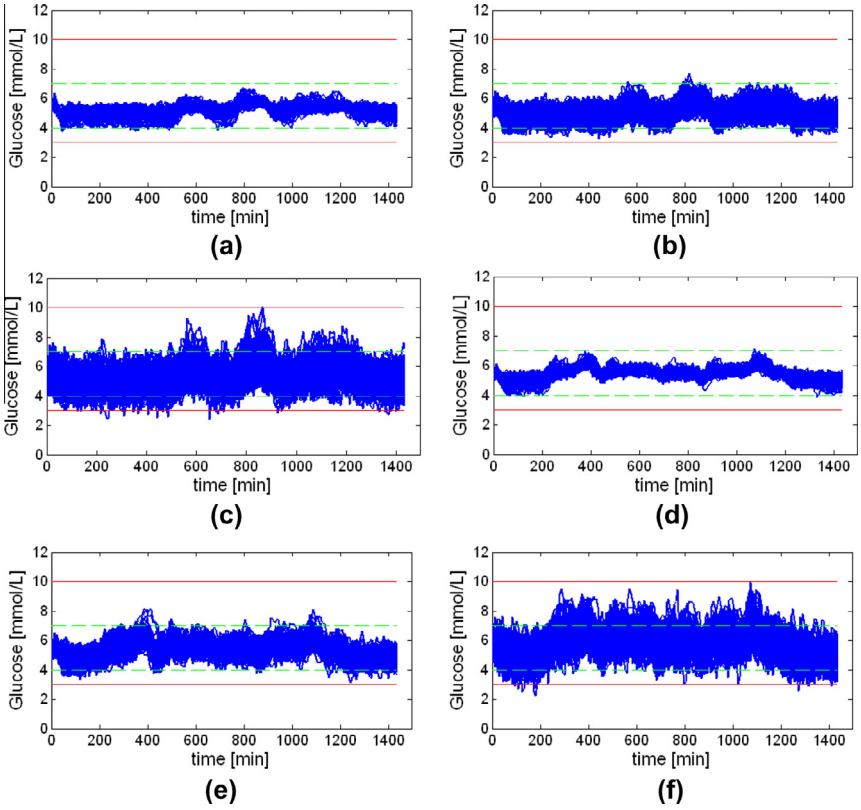

Fig. 12. Test experiments. Each panel shows 125 resulting glucose profiles for simulated patients with variability parameter σ_{ito} of 0.15, 0.25 and 0.50 under both meal routines I and II. (a) Results for $\sigma_{ito} = 0.15$ and meal routine I; (b) Results for $\sigma_{ito} = 0.25$ and meal routine I; (c) Results for $\sigma_{ito} = 0.50$ and meal routine I (d) Results for $\sigma_{ito} = 0.15$ and meal routine II; (e) Results for $\sigma_{ito} = 0.25$ and meal routine II; (f) Results for $\sigma_{ito} = 0.50$ and meal routine II.

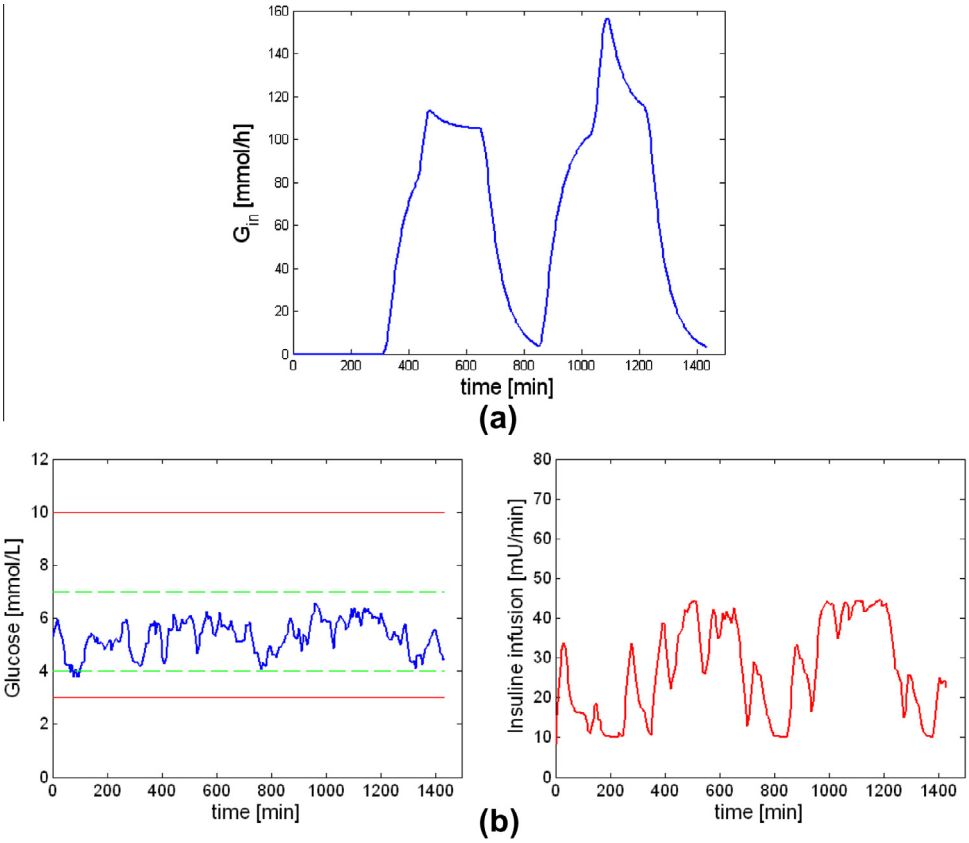

Fig. 13. Testing case. (a) Rate of glucose absorption from the gut of a testing meal routine; (b) Results of controlling the patient under the testing meal routine.

To apply the RL framework, a remaining issue is to define the variables that made up the vector which provides a perception of the system state. To this aim, the measured blood glucose concentration at time t (G_t) and the insulin flow rate in the previous time step D_{t-1} are used to characterize the glycemic state of the patient at time t. In this way, a perception of the system state is conveniently defined through $\mathbf{x}_t = (G_t, D_{t-1})^T$. Furthermore, the scalar control action, u_t, is chosen as the change to the insulin infusion rate. An advantage of perceiving the system state $\mathbf{x}_t$ in this way is that it only involves readily known variables, yet they are informative enough to describe the physiological state of a patient for successfully controlling blood glucose.

4.2. Obtaining a generic control policy in off line way

A generic control policy for glucose regulation was obtained by simulation-based learning using the model-based algorithm presented in Fig. 1 (Algorithm 1). Patient responses to control actions were simulated using the model presented in Section 2.4 with parameters in Table 1. To assess Algorithm 1, different meal routines were considered to obtain the corresponding generic control policies. The dynamic effect of a meal on blood glucose behavior mainly depends of its carbohydrate contents. Two feeding schedules indicated in Table 2 which are significantly different regarding the carbohydrate intakes are considered. In Fig. 10, the effect of carbohydrate ingestions of each meal routine (given in Table 2) in terms of the glucose absorption from the gut G_{in} are shown. A meal can be understood as an external disturbance that causes blood glucose perturbations and it is useful to interpret the results. It can be seen that the Routine II is significantly more intensive in term of carbohydrate intakes.

The glucose profiles resulting of applying the optimal policies found using Algorithm 1 for the same patient (simulated) under meal routines I and II, respectively, are given in Fig. 11a and b. In

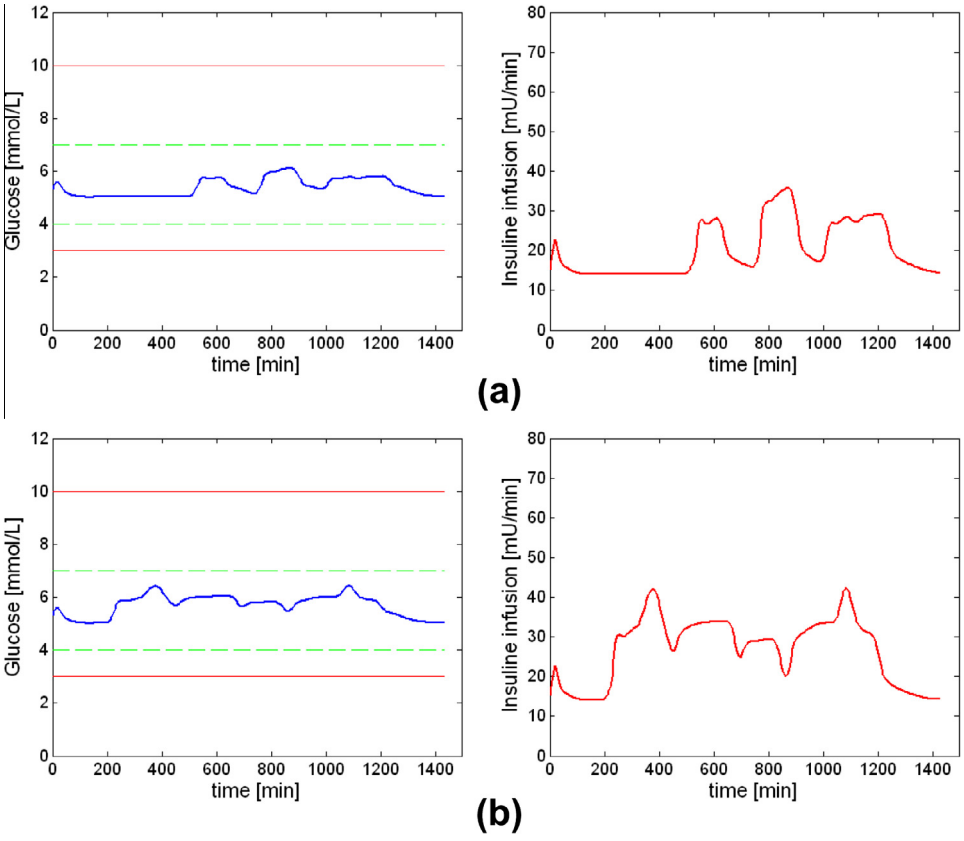

Fig. 14. Deterministic case. (a) Results for $\sigma_{ito} = 0.0$ and meal routine I; (b) Results for $\sigma_{ito} = 0.0$ and meal routine II.

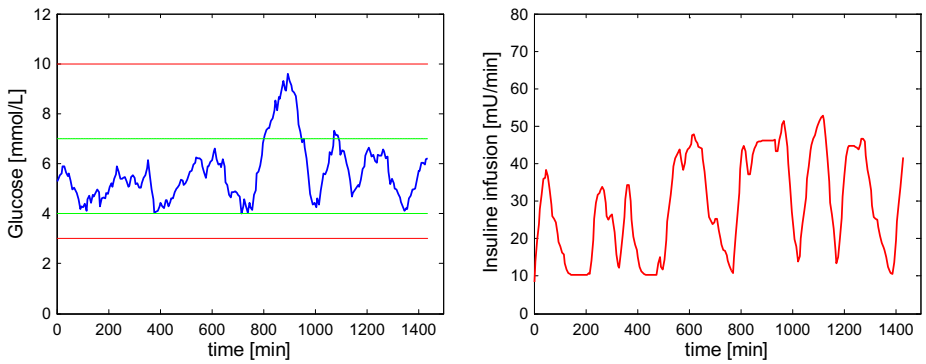

Fig. 15. Performance of π^* as initial policy for a specific patient under meal routine I.

this case, the algorithm inputs are: $\mathcal{X}_0 = \{\mathbf{x}_0^l\}_{l=1}^{n_0}$ with $n_0 = 5$ where $\mathbf{x}_0^l = (G_l D_l)$ such that G_l and D_l are random values from intervals $G_l \in [3,10]$ and $D_l \in [0,80]$; $\delta = 10\%$, $\gamma = 0.95$ and π_0 is a random policy. The changes in insulin infusion rate are performed every 6 min, whereby $\mathcal{T} = 240$ is used to ensure full-day trajectories.

Results depicted in Fig. 11 clearly indicate that control actions needed to regulate the blood glucose levels for the Routine II is more demanding in terms of the exogenous insulin required. However, in both cases the time profile of insulin infusion required is vividly correlated with the glucose absorption rate. Glycemic

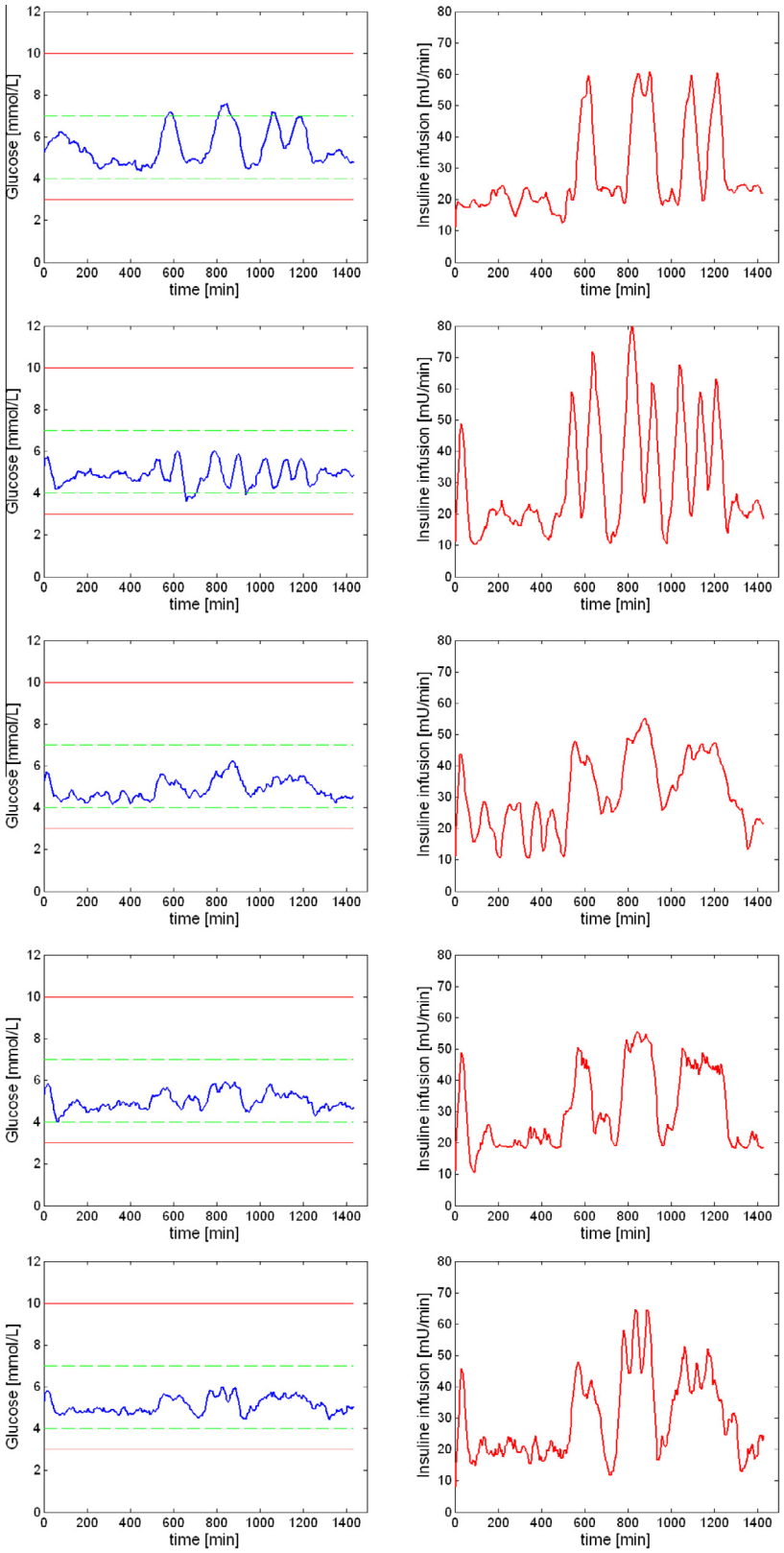

Fig. 16. Results of five days of a policy personalization carried out by Algorithm 4, starting from π^*, for a (virtual) patient under the meal routine I.

variability for both meal routines is significantly low and both control policies exhibit outstanding performance. It is worth noting that for different carbohydrate intakes, the blood glucose concentration is always confined within the green dot lines as can be expected for an optimal control policy.

The flexibility of modeling the policies by GP models allows combining the support sets of different generic policies, which are obtained in a separated way through different simulation experiments (see Section 2.3). In this way, we obtain a new generic control policy π^* based on an augmented support data set which includes the support data of policies obtained for meal routine I and II. Thus, the new generic policy will have an outstanding generalization ability to deal with different situations that a patient may experience. We test this new policy for meal routines in Table 2. In Fig. 12, different test experiments are shown. Each experiment includes 125 simulations for a patient (virtual) under a meal routine (I or II) considering different degrees of variability in glycemic behavior, which is established by the σ_{ito} parameter in the model discussed in Section 2.4.

Fig. 13 shows a situation where there exists quite high carbohydrate intakes spaced in time (see Fig. 13(a)). Fig. 13(b) exhibits the corresponding result for this testing case. Results obtained demonstrate that the obtained generic control policy can successfully control glycemic variability while maintaining it within the safe range (soft thresholds). In Fig. 14, the extreme situation where the patient follows a deterministic behavior is shown. It can be seen that the policy achieves an optimal performance for meal routine I (Fig. 14(a)) and routine II (Fig. 14(b)). Note the control problem becomes rather simple when the policy obtained using Algorithm 1 is applied. In the next section, the generic policy π^* is taken as the initial policy which will be personalized in real-time to a certain patient using the Algorithm 4.

4.3. Real time personalization of a generic control policy via on-line interactions

The main difference between off-line learning and on-line adaptation is that in the latter, the policy is modified based on the data actually generated in real-time by online interactions between the patient and the artificial pancreas which implements the Algorithm 4. To illustrate on-line policy adaptation, the generic policy π^* is taken as the starting point and then increasingly personalized to a specific patient upon data obtained from applying π^*.

As a proxy for a real patient, the model described in Section 2.4 is parameterized such that the policy π^* used do not have an optimal performance. Thus, setting $Sh = 0.28$ (hepatic sensitivity parameter) and considering the meal routine I (Table 2) when applying $\pi^*_{c_0} = \pi^*$ to the virtual patient, it can be seen in the results shown in Fig. 15 that this initial policy ($\pi^*_{c_0}$) exhibits an acceptable performance, but there is plenty of room for performance improving through real-time personalization.

Results obtained for the personalization strategy using the Algorithm 4 during five days (top-down) are summarized in Fig. 16. The algorithm inputs are: the generic policy π^* with its support data set $\mathbb{D}, \eta = 1.0e^{-04}, h = 8000, N_{max} = 800, \rho = 1, \beta = 2, \gamma = 0.95$ and the reward function $r(\cdot)$ is the one described in Section 4.1. Each panel in Fig. 16 corresponds to a daily run of Algorithm 4. It is rather remarkable the fast evolution from a safe, yet sub-optimal generic policy π^* to an optimal personalized control policy π^*_c. It can be seen that after the third day of policy personalization, only minor changes are perceived in the patient's glycemic variability when the artificial pancreas implements the adapted control policy.

Now, let us assume that the same (virtual) patient suddenly changes his diet after the day #5 and adopts the meal routine II. Results obtained when the patient is controlled using the current policy $\pi^*_{c_5}$ are shown in Fig. 17 (note that on-line policy adaptation is not yet carried out). As can be seen, the version $\pi^*_{c_5}$ of the specialized policy is underperforming to manage the glucose variability of the patient (virtual). To adapt the policy to the new feeding pattern, the Algorithm 4 is applied. In Fig. 18, results obtained during on-line policy learning over five consecutive days are shown. Similarly to Fig. 16, each panel of Fig. 18 shows a daily run (top-down). The improvement of the policy as the adaptable artificial pancreas interacts with the patient is quite remarkable.

Again, let us assume that the patient suddenly changes his feeding routine such that carbohydrate intakes follow the profile shown in Fig. 19a. If the control policy $\pi^*_{c_{10}}$ is applied results obtained are given in Fig. 19b, which exhibits that the artificial pancreas is underperforming. Again, if on-line policy learning is carried out over the next consecutive days, optimal control of glycemic variability is restored (see Fig. 20).

5. Finals remarks

A strategy for on-line learning and real-time adaptation of a control policy for personalization of an artificial pancreas to a specific patient has been presented. This development is an important step forward towards an individualized therapy in diabetes management. Unlike other adaptive approaches, policy personalization proposes a promissory alternative which allows a real-time continuous interaction with the patient whilst critical patient data are used to update on-line the model of a specific patient dynamics. This enables that the control policy can be adapted on the fly according to a given patient metabolism and lifestyle. By considering patient-specific data and working with continuous data stream without requiring any state discretization, the on-line personalization policy can be applied regardless which control strategy was used to define the initial policy. That is, the initial policy would

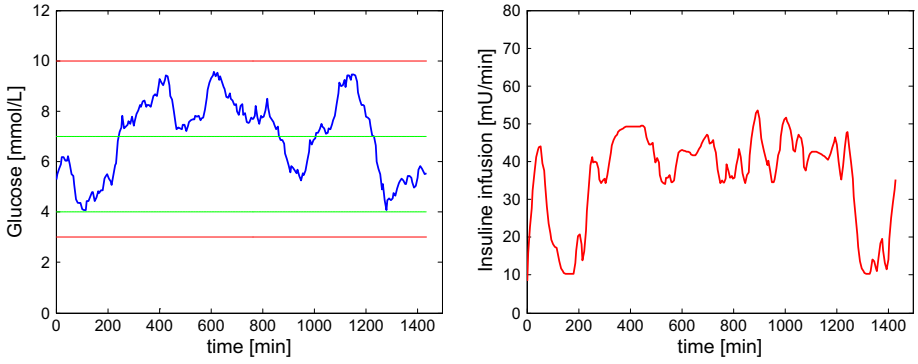

Fig. 17. Performance of $\pi^*_{c_5}$ for a specific patient under meal routine II.

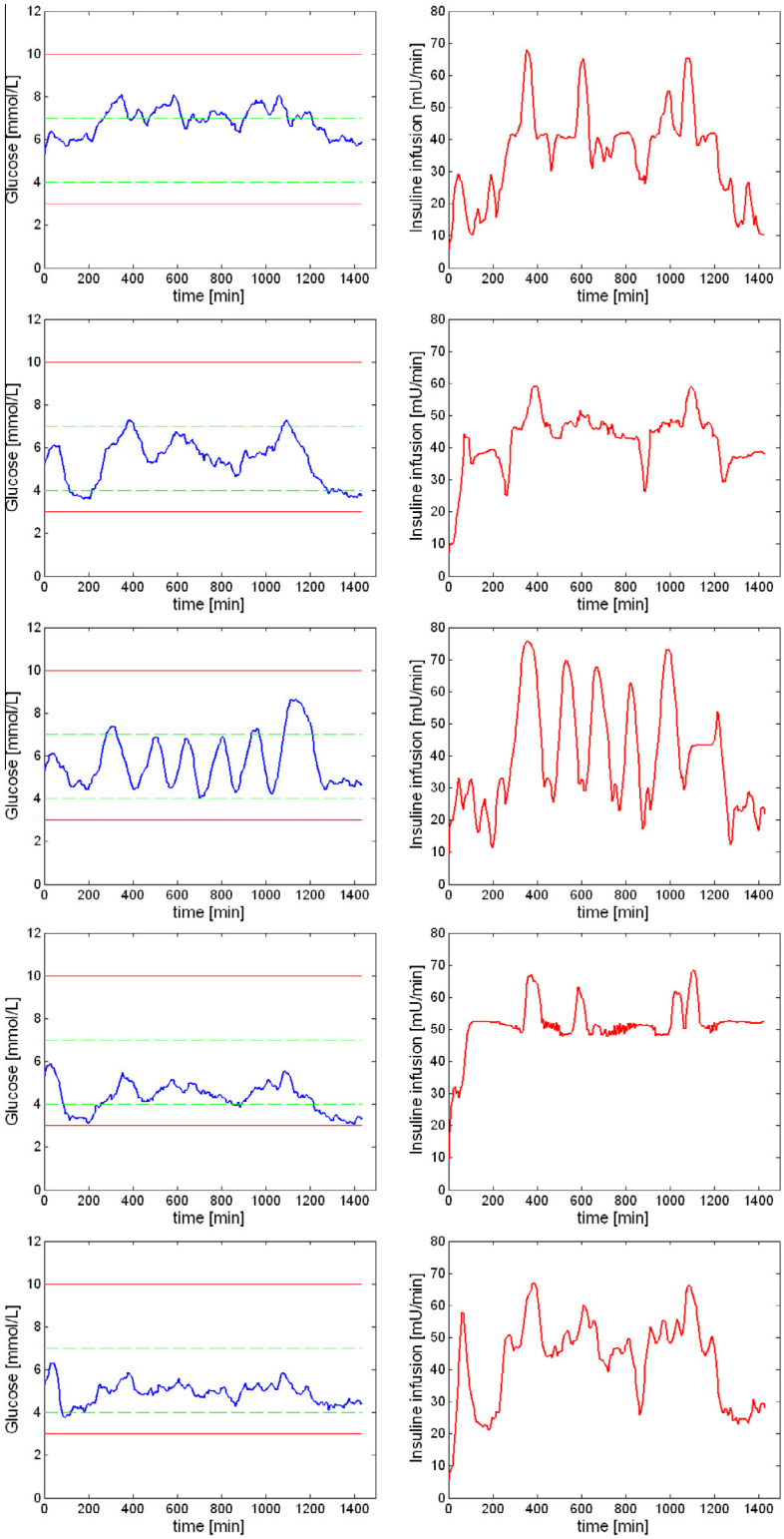

Fig. 18. Results of the next five days of a policy personalization carried out by Algorithm 4, starting from $\pi^{*}_{c_5}$, for a virtual patient under the meal routine II.

be obtained using reinforcement learning or mp-MPC. However, Bayesian active learning algorithm and the sparsification algorithm play a key role in favoring our RL formulation. Moreover, a compact representation of the control policy modeled based on GPs which greatly facilitates real-time computing is proposed. This is the main advantage of the proposed approach for its implementation in wearable devices.

Blood glucose control is intrinsically a regulation problem for which we have proposed our approach based on the RL framework unlike the proposal made in De Paula and Martínez (2012b) which belongs to the field of dynamic programming and was developed to solve mainly an episodic learning task in a multi-modal setting. Also, in De Paula and Martínez (2012b) learning is based on Lebesgue sampling due to control modes whereas in the present work

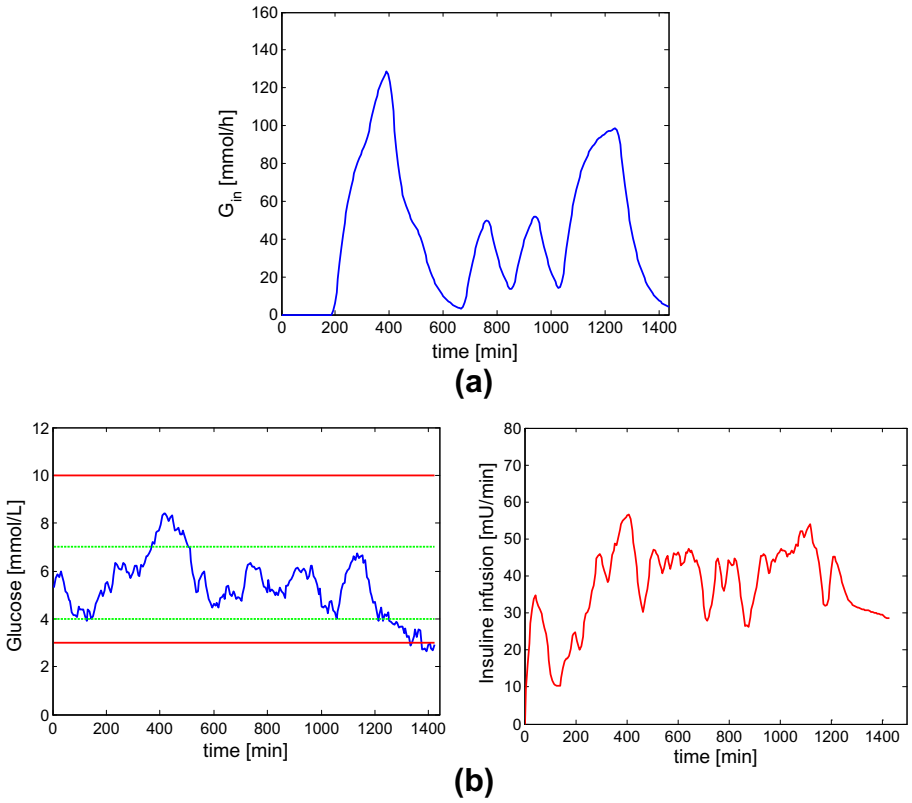

Fig. 19. Change in feeding habits. (a) Rate of glucose absorption from the gut for the new diet; (b) Performance of $\pi^*_{c_{10}}$ for the virtual patient under the new feeding routine.

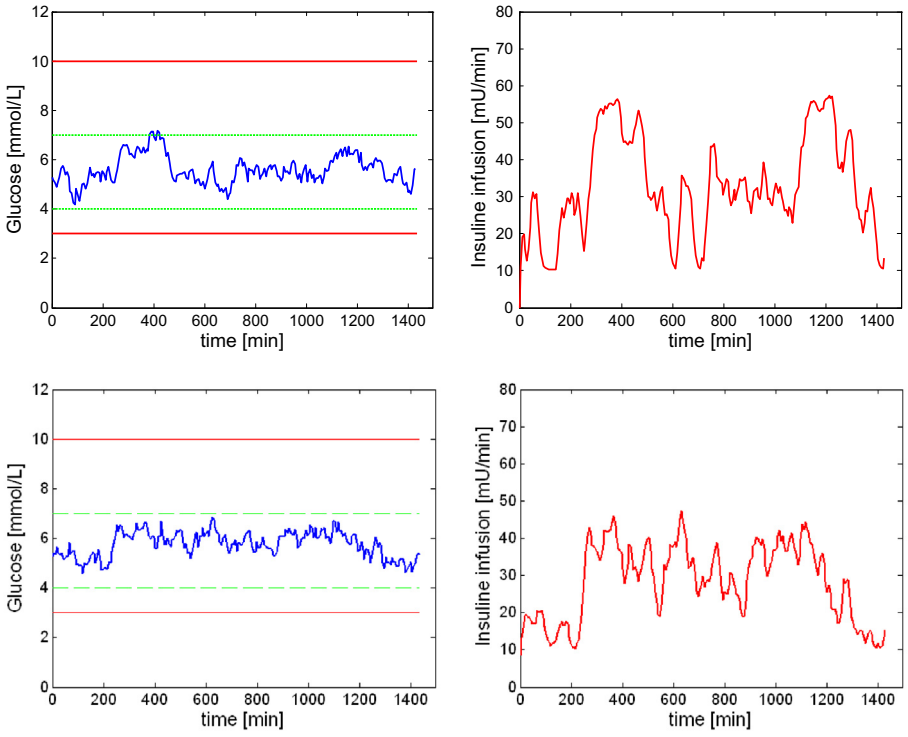

Fig. 20. Results of the next two days of a policy personalization carried out by Algorithm 4, starting from $\pi^*_{c_{10}}$, for a virtual patient under the latter meal routine.

Riemann sampling is used. The Bayesian active learning strategy in our previous work (De Paula & Martínez, 2012b) was developed for a simulation-based schema which is not apt enough for policy adaptation in real-time interactions. Therefore, in the present work an improved BAL algorithm capable of interacting in real-time with a certain patient is proposed. Also, the present development includes a sparsification algorithm that allows working with a limited number of data for data-driven modeling of the glucose dynamics. These features makes policy adaptation a computational tractable problem which favor working in real-time. Unlike an

expert system, policy personalization is carried out without any prior expert knowledge beyond the initial control policy used as input to the algorithm.

One of the major limitations of the research work made so far lies in resorting to *in silico* patient experimentation for assessing policy personalization. Therefore is imperative to test policy adaptation on real patients, or animals, for which is mandatory working as part of a multidisciplinary team. In turn, this limitation encourages a direction for our future work. On the other hand, the RL ensures convergence even under conditions for constrained action selection (Sutton & Barto, 1998). Therefore, it is interesting to consider the possibility of enhancing our proposal by accommodating expert knowledge to make sense of the personalized policy so that it can be better understood by physicians, and on that basis safety constraints can be added.

Current research efforts focus on three avenues. First, we are working towards integrating the personalized policies of patients in a homogeneous group in a "grand" policy which can be used as a robust initial policy for a new patient in the same group. Secondly, the ambitious idea of an autonomic artificial pancreas is being pursued. In this regard, self-optimization (policy personalization) is just one of the key functions. Also, self-monitoring and self-diagnostic functionalities are being developed. Finally, prototyping policy personalization so it can be readily used in tablets and smart-phones is an important objective for wide dissemination of the policy personalization idea.

Acknowledgement

The authors would like to thanks the National Research and Technology Council of Argentina (CONICET) for the economic support (PIP 2010 0386) and for providing Mariano De Paula with a doctoral fellowship.

Appendix A. Calculation when augmenting a data set

Whenever a support set $\mathbb{D}$ is augmented by adding a new data point $\mathscr{d}_{new}$, the linear independence measures δ^i for all $\mathscr{d}_i \in \mathbb{D}$ must be updated (Engel et al., 2004). This involves updating $\boldsymbol{a}^i$,$\boldsymbol{K}^{i-1}$ and $\boldsymbol{k}^i$ for every *i-th* point. Thus, the K^i must be extended by a row/column and $\boldsymbol{k}^i$ by a single value, such that

$$\boldsymbol{K}^i_{new} = \begin{bmatrix} \boldsymbol{K}^i_{old} & \boldsymbol{k}_{m+1} \\ \boldsymbol{k}^T_{m+1} & k_{m+1} \end{bmatrix} \tag{A.1}$$

$$\boldsymbol{k}^i_{new} = \begin{bmatrix} \boldsymbol{k}^i_{old} & \boldsymbol{k}_{i,m+1} \end{bmatrix}^T \tag{A.2}$$

where $k_{m+1} = \mathrm{k}(\mathscr{d}_{new}, \mathscr{d}_{new}), \boldsymbol{k}_{i,m+1} = \mathrm{k}(\mathscr{d}_i, \mathscr{d}_{new}), \boldsymbol{k}_{new} = \mathrm{k}(\mathbb{D}^i, \mathscr{d}_{new})$ with $\mathbb{D}^i = \mathbb{D} \setminus \{\mathscr{d}_i\}$.

With the Eqs. (A.1) and (A.2), the inverse kernel matrix $(\boldsymbol{K}^i_{new})^{-1}$ is given by

$$\left(\boldsymbol{K}^i_{new}\right)^{-1} = \frac{1}{\gamma_i} \begin{bmatrix} \gamma_i \left(\boldsymbol{K}^i_{old}\right)^{-1} + \alpha_i \alpha_i^T & -\alpha_i \\ -\alpha_i^T & 1 \end{bmatrix} \tag{A.3}$$

Then, the independence measure δ^i for all point $\mathscr{d}_i \in \mathbb{D}$ is given by

$$\delta^i = \mathrm{k}(\mathscr{d}_i, \mathscr{d}_i) - \left(\boldsymbol{k}^i_{new}\right)^T \boldsymbol{a}^i_{new} \tag{A.4}$$

being

$$\boldsymbol{a}^i_{new} = \frac{1}{\gamma_i} \begin{bmatrix} \gamma_i \boldsymbol{a}^i_{new} + \boldsymbol{\alpha}_i \boldsymbol{\alpha}_i^T \boldsymbol{k}^i_{old} - k_{i,m+1} \boldsymbol{\alpha}_i \\ -\boldsymbol{\alpha}_i^T \boldsymbol{k}^i_{old} + k_{i,m+1} \end{bmatrix} \tag{A.5}$$

where $\boldsymbol{\alpha}_i = (\boldsymbol{K}^i_{old})^{-1} \boldsymbol{k}_{m+1}$ y $\gamma_i = k_{m+1} - \boldsymbol{k}^T_{m+1} \boldsymbol{\alpha}_i$.

Appendix B. Calculation when replacing an old data by a new one in a data set

Every time that a *j*th data point $\mathscr{d}_j$ is replaced by a new data $\mathscr{d}_{new}$ in $\mathbb{D}$, the independence measure δ^i for all point $\mathscr{d}_i \in \mathbb{D}$ must be updated (Nguyen-Tuong & Peters, 2011a). This involves a manipulation of the *j*th row/column of $\boldsymbol{K}^i$ and the *j*th value of $\boldsymbol{k}$, this is

$$\boldsymbol{K}^i_{new} = \begin{bmatrix} \boldsymbol{K}^i_{old(1:j)} & \boldsymbol{k}_{m+1(1:j)} & \boldsymbol{K}^i_{old(j:m)} \\ \boldsymbol{k}^T_{m+1(1:j)} & \boldsymbol{k}_{m+1} & \boldsymbol{k}^T_{m+1(j:m)} \\ \boldsymbol{K}^{iT}_{old(j:m)} & \boldsymbol{k}_{m+1(j:m)} & \boldsymbol{K}^{iT}_{old(1:j)} \end{bmatrix} \tag{B.1}$$

$$\boldsymbol{k}^i_{new} = \begin{bmatrix} \boldsymbol{k}^i_{old(1:j)} & k_{i,m+1} & \boldsymbol{k}^i_{old(j:m)} \end{bmatrix}^T \tag{B.2}$$

where $k_{m+1} = \mathrm{k}(\mathscr{d}_{new}, \mathscr{d}_{new})$, $k_{i,m+1} = \mathrm{k}(\mathscr{d}_i, \mathscr{d}_{new}), \boldsymbol{k}_{new} = \mathrm{k}(\mathbb{D}^i, \mathscr{d}_{new})$. Then, the independence measures $\delta^i \forall \mathscr{d}_i \in \mathbb{D}$ must be updated as in Eq. (A.4), where

$$\boldsymbol{a}^i_{new} = (\boldsymbol{K}^i_{new})^{-1} \boldsymbol{k}^i_{new} \tag{B.3}$$

while $\left(\boldsymbol{K}^i_{new}\right)^{-1}$ can be recalculated using the following update rule

$$\left(\boldsymbol{K}^i_{new}\right)^{-1} = \boldsymbol{A}^* - \frac{\mathrm{row}_j[\boldsymbol{A}^*]^T \boldsymbol{r}^T \boldsymbol{A}^*}{1 + \boldsymbol{r}^T \mathrm{row}_j[\boldsymbol{A}^*]^T} \tag{B.4}$$

where

$$\boldsymbol{A}^* = \boldsymbol{A}^* - \frac{(\boldsymbol{K}^i_{old})^{-1} \boldsymbol{r}\ \mathrm{row}_j\left[(\boldsymbol{K}^i_{old})^{-1}\right]}{1 + \boldsymbol{r}^T \mathrm{row}_j\left[\left(\boldsymbol{K}^i_{old}\right)^{-1}\right]^T} \tag{B.5}$$

where $\boldsymbol{r} = \boldsymbol{k}_{m+1} - \mathrm{row}_j\left[\boldsymbol{K}^i_{old}\right]^T$ and $\mathrm{row}_j[\boldsymbol{M}]$ denotes the *j*th row of a given matrix $\boldsymbol{M}$.

Appendix C. Predictions with uncertain inputs

In the following we refer to the results given in Deisenroth et al. (2009) of how to predict with GPs for uncertain inputs. Considerer the problem of predicting a function value $h(\mathbf{x}^*)$ for an uncertain test input which is Gaussian distributed $\mathbf{x}^* \sim \mathcal{N}(\mu, \Sigma)$, where $h \sim GP$ with a square exponential (SE) covariance function $\mathrm{k}(\cdot,\cdot)$ (as in Eq. (8)), correspond to seeking the exact predictive distribution:

$$p(h(\mathbf{x}^*)|\mu, \Sigma) = \int p(h(\mathbf{x}^*)|\mathbf{x}^*) p(\mathbf{x}^*|\mu, \Sigma) d\mathbf{x}^* \tag{C.1}$$

which is not a Gaussian distribution. The mean and the variance of the predictive distribution $p(h(\mathbf{x}^*)|\mu,\Sigma)$ in Eq. (C.1) are given by Eqs. (6) and (7), respectively. When a GP model regards a square exponential (SE) kernel, $\mathrm{k}_{\mathrm{SE}}(\cdot,\cdot)$, the mean μ^* and the variance $\mathcal{V}^2$ of predictive distribution in Eq. (C.1) can be computed in close form. Approximating the exact predictive distribution with a Gaussian, which possesses the same mean and variance, the mean μ^* is given by:

$$\begin{aligned} \mu^* &= E_h[E_{\mathbf{x}^*}[h(\mathbf{x}^*)]] = E_{\mathbf{x}^*}[E_h[h(\mathbf{x}^*)]] \\ &= E_{\mathbf{x}^*}[m(\mathbf{x}^*)] = \int m(\mathbf{x}^*) p(\mathbf{x}^*) d\mathbf{x}^* = \boldsymbol{\beta}^{\mathrm{T}} \boldsymbol{l} \end{aligned} \tag{C.2}$$

with $\boldsymbol{\beta} = (\boldsymbol{K} + \sigma_w^2 \boldsymbol{I})^{-1} \boldsymbol{Y}$, where

$$\begin{aligned} l_i &= \int Cov(\mathbf{x}_i, \mathbf{x}^*) p(\mathbf{x}^*) d\mathbf{x}^* \\ &= \alpha^2 |\Sigma \Lambda^{-1} + I|^{-1/2} \exp\left(-\frac{1}{2}(\mathbf{x}_i - \mu)^{\mathrm{T}} (\Sigma + \Lambda^{-1})(\mathbf{x}_i - \mu)\right) \end{aligned}$$

here, Λ is a diagonal matrix with the characteristic length-scales. The variance $\mathcal{V}^2$ of the predictive distribution of Eq. (C.1) is given by:

$$\begin{aligned}\mathcal{V}^2 &= E_{\mathbf{x}^*}[m(\mathbf{x}^*)^2] + E_{\mathbf{x}^*}[Cov(\mathbf{x}^*,\mathbf{x}^*)^2] - E_{\mathbf{x}^*}[m(\mathbf{x}^*)]^2 \\ &= \boldsymbol{\beta}^{\mathrm{T}}\boldsymbol{L}\boldsymbol{\beta} + \alpha^2 - tr((\mathbf{K}+\sigma_w^2\boldsymbol{I})^{-1}\boldsymbol{L}) - \mu^{*2}\end{aligned} \tag{C.3}$$

where

$$L_{ij} = \frac{Cov(\mathbf{x}_i,\mu)Cov(\mathbf{x}_j,\mu)}{|2\Sigma\boldsymbol{\Lambda}^{-1}+I|^{-1/2}} \exp\left[(\tilde{\boldsymbol{z}}_{ij}-\mu)^T\left(\Sigma+\frac{1}{2}\boldsymbol{\Lambda}\right)^{-1}\Sigma\boldsymbol{\Lambda}^{-1}(\tilde{\boldsymbol{z}}_{ij}-\mu)\right]$$

with $\tilde{\boldsymbol{z}}_{ij} = \frac{1}{2}(\mathbf{x}_i + \mathbf{x}_j)$. Note, that the predictive mean μ^* and the predictive variance $\mathcal{V}^2$ depend explicitly on the mean, μ, and covariance matrix, Σ, of the uncertain input $\mathbf{x}^*$.

References

Akbari Torkestani, J., & Ghanaat Pisheh, E. (2014). A learning automata-based blood glucose regulation mechanism in type 2 diabetes. *Control Engineering Practice, 26*, 151–159.

Baranes, A., & Oudeyer, P.-Y. (2013). Active learning of inverse models with intrinsically motivated goal exploration in robots. *Robotics and Autonomous Systems, 61*(1), 49–73.

Bequette, B. W. (2005). A critical assessment of algorithms and challenges in the development of a closed-loop artificial pancreas. *Diabetes Technology & Therapeutics, 7*(1), 28–47.

Bequette, B. W. (2012). Challenges and recent progress in the development of a closed-loop artificial pancreas. *Annual Reviews in Control, 36*(2), 255–266.

Bergman, R. N., Ider, Y. Z., Bowden, C. R., & Cobelli, C. (1979). Quantitative estimation of insulin sensitivity. *The American Journal of Physiology, 236*(6), 667–677.

Bergman, R. N., Phillips, L. S., & Cobelli, C. (1981). Physiologic evaluation of factors controlling glucose tolerance in man: measurement of insulin sensitivity and beta-cell glucose sensitivity from the response to intravenous glucose. *Journal of Clinical Investigation, 68*(6), 1456–1467.

Bothe, M. K., Dickens, L., Reichel, K., Tellmann, A., Ellger, B., Westphal, M., et al. (2013). The use of reinforcement learning algorithms to meet the challenges of an artificial pancreas. *Expert Review of Medical Devices, 10*(5), 661–673.

Buckingham, B., Caswell, K., & Wilson, D. M. (2007). Real-time continuous glucose monitoring. *Current Opinion in Endocrinology, Diabetes and Obesity, 14*(4), 288–295.

Chee, F., Fernando, T. L., Savkin, A. V., & van Heeden, V. (2003). Expert PID control system for blood glucose control in critically ill patients. *IEEE Transactions on Information Technology in Biomedicine, 7*(4), 419–425.

Chen, X., Gao, Y., & Wang, R. (2013). Online selective Kernel-based temporal difference learning. *IEEE Transactions on Neural Networks and Learning Systems, 24*(12), 1944–1956.

Cobelli, C., Renard, E., & Kovatchev, B. (2011). Artificial Pancreas: Past, Present, Future. *Diabetes, 60*(11), 2672–2682.

Cohn, D. A., Ghahramani, Z., & Jordan, M. I. (1996). Active Learning with statistical models. *Journal of Artificial Intelligence Research, 4*, 129–145.

Cosenza, B. (2012). Off-line control of the postprandial glycemia in type 1 diabetes patients by a fuzzy logic decision support. *Expert Systems with Applications, 39*(12), 10693–10699.

Daskalaki, E., Diem, P., & Mougiakakou, S. G. (2013a). An Actor–Critic based controller for glucose regulation in type 1 diabetes. *Computer Methods and Programs in Biomedicine, 109*(2), 116–125.

Daskalaki, E., Diem, P., & Mougiakakou, S. G. (2013). Personalized tuning of a reinforcement learning control algorithm for glucose regulation. In *2013 35th Annual international conference of the IEEE engineering in medicine and biology society (EMBC)* (pp. 3487–3490).

Daskalaki, E., Prountzou, A., Diem, P., & Mougiakakou, S. G. (2012). Real-time adaptive models for the personalized prediction of glycemic profile in type 1 diabetes patients. *Diabetes Technology & Therapeutics, 14*(2), 168–174.

Deisenroth, M. P. (2010). *Efficient reinforcement learning using gaussian processes*. KIT Scientific Publishing.

Deisenroth, M. P., Rasmussen, C. E., & Peters, J. (2009). Gaussian process dynamic programming. *Neurocomputing, 72*(7-9), 1508–1524.

De Paula, M., & Martínez, E. (2012). Probabilistic optimal control of blood glucose under uncertainty. In *22nd European symposium on computer aided process engineering* (Vol. 30, pp. 1357–1361).

De Paula, M., & Martínez, E. C. (2012b). Optimal operation of discretely controlled continuous systems under uncertainty. *Industrial & Engineering Chemistry Research, 51*(42), 13743–13764.

Dua, P., Doyle, F. J., 3rd, & Pistikopoulos, E. N. (2006). Model-based blood glucose control for Type 1 diabetes via parametric programming. *IEEE Transactions on Bio-Medical Engineering, 53*(8), 1478–1491.

Dua, P., Kouramas, K., Dua, V., & Pistikopoulos, E. N. (2008). MPC on a chip–Recent advances on the application of multi-parametric model-based control. *Computers & Chemical Engineering, 32*(4-5), 754–765.

Engel, Y., Mannor, S., & Meir, R. (2004). The kernel recursive least-squares algorithm. *IEEE Transactions on Signal Processing, 52*(8), 2275–2285.

Eren-Oruklu, M., Cinar, A., Quinn, L., & Smith, D. (2009). Adaptive control strategy for regulation of blood glucose levels in patients with type 1 diabetes. *Journal of Process Control, 19*(8), 1333–1346.

Esfandiari, N., Babavalian, M. R., Moghadam, A.-M. E., & Tabar, V. K. (2014). Knowledge discovery in medicine: Current issue and future trend. *Expert Systems with Applications, 41*(9), 4434–4463.

Favero, S. D., Bruttomesso, D., Palma, F. D., Lanzola, G., Visentin, R., Filippi, A., et al. (2014). First Use of Model Predictive Control in Outpatient Wearable Artificial Pancreas. *Diabetes Care, 37*(5), 1212–1215.

Gantt, J. A., Rochelle, K. A., & Gatzke, E. P. (2007). Type 1 diabetic patient insulin delivery using asymmetric PI control. *Chemical Engineering Communications, 194*(5), 586–602.

García, C. E., Prett, D. M., & Morari, M. (1989). Model predictive control: Theory and practice—A survey. *Automatica, 25*(3), 335–348.

Gijsberts, A., & Metta, G. (2013). Real-time model learning using Incremental Sparse Spectrum Gaussian Process Regression. *Neural Networks, 41*, 59–69.

Grosman, B., Dassau, E., Zisser, H. C., Jovanovic, L., & Doyle, F. J. (2010). Zone model predictive control: A strategy to minimize hyper- and hypoglycemic events. *Journal of Diabetes Science and Technology, 4*(4), 961–975.

Guyton, A. C., & Hall, J. E. (2000). *Textbook of medical physiology*. Saunders.

Hanaire, H., Lassmann-Vague, V., Jeandidier, N., Renard, E., Tubiana-Rufi, N., Vambergue, A., et al. (2008). Treatment of diabetes mellitus using an external insulin pump: the state of the art. *Diabetes & Metabolism, 34*(4, Supplement 1), 401–423.

Heinemann, L. (2002). Variability of insulin absorption and insulin action. *Diabetes Technology & Therapeutics, 4*(5), 673–682.

Hester, T., Quinlan, M., & Stone, P. (2012). RTMBA: A real-time model-based reinforcement learning architecture for robot control. In *2012 IEEE international conference on robotics and automation (ICRA)* (pp. 85–90).

Hovorka, R., Canonico, V., Chassin, L. J., Haueter, U., Massi-Benedetti, M., Federici, M. O., et al. (2004). Nonlinear model predictive control of glucose concentration in subjects with type 1 diabetes. *Physiological Measurement, 25*(4), 905.

Ito, K. (1951). *On stochastic differential equations*. Memoirs of The American Mathematical Society.

Krause, A., & Guestrin, C. (2007). Nonmyopic active learning of Gaussian processes: an exploration–exploitation approach. In *Proceedings of the 24th International Conference on Machine Learning (ICML-07)* (pp. 449–456).

Krüger, L., Slabber, M., Joubert, G., Venter, C. S., & Vorster, H. H. (2007). Intra- and inter-individual variation in blood glucose response to white bread and glucose in patients with type 2 diabetes mellitus. South African Journal of Clinical Nutrition.

Lee, H., Buckingham, B. A., Wilson, D. M., & Bequette, B. W. (2009). A closed-loop artificial pancreas using model predictive control and a sliding meal size estimator. *Journal of Diabetes Science and Technology, 3*(5), 1082–1090.

Lehmann, E. D., & Deutsch, T. (1992). A physiological model of glucose–insulin interaction in type 1 diabetes mellitus. *Journal of Biomedical Engineering, 14*(3), 235–242.

Lunze, K., Singh, T., Walter, M., Brendel, M. D., & Leonhardt, S. (2013). Blood glucose control algorithms for type 1 diabetic patients: A methodological review. *Biomedical Signal Processing and Control, 8*(2), 107–119.

MacKay, D. J. C. (2003). *Information theory, inference, and learning algorithms*. Cambridge, UK; New York: Cambridge University Press.

Magni, L., Raimondo, D. M., Bossi, L., Man, C. D., De Nicolao, G., Kovatchev, B., et al. (2007). Model predictive control of type 1 diabetes: An in silico trial. *Journal of Diabetes Science and Technology (Online), 1*(6), 804–812.

Magni, L., Raimondo, D. M., Dalla Man, C., De Nicolao, G., Kovatchev, B., & Cobelli, C. (2009). Model predictive control of glucose concentration in type I diabetic patients: An in silico trial. *Biomedical Signal Processing and Control, 4*(4), 338–346.

Marchetti, G., Barolo, M., Jovanovic, L., Zisser, H., & Seborg, D. E. (2008). An Improved PID Switching Control Strategy for Type 1 Diabetes. *IEEE Transactions on Biomedical Engineering, 55*(3), 857–865.

Markakis, M. G., Mitsis, G. D., Papavassilopoulos, G. P., & Marmarelis, V. Z. (2010). Nonparametric modeling and model-based control of the insulin–glucose system. New developments in biomedical engineering, InTech.

Marling, C. R., Shubrook, J. H., Vernier, S. J., Wiley, M. T., & Schwartz, F. L. (2011). Characterizing blood glucose variability using new metrics with continuous glucose monitoring data. *Journal of Diabetes Science and Technology, 5*(4), 871–878.

McCausland, L., Mareels, I. M. Y., Barnett, R. W., & Arad, D. (1999). An adaptive expert system for blood glucose control in type 1 diabetes mellitus. *Engineering in Medicine and Biology, 2*, 1209.

Nguyen-Tuong, D., & Peters, J. (2011a). Incremental online sparsification for model learning in real-time robot control. *Neurocomputing, 74*(11), 1859–1867.

Nguyen-Tuong, D., & Peters, J. (2011b). Model learning for robot control: a survey. *Cognitive Processing, 12*(4), 319–340.

Palerm, C. C., Zisser, H., Jovanovič, L., & Doyle, F. J. III, (2008). A run-to-run control strategy to adjust basal insulin infusion rates in type 1 diabetes. *Journal of Process Control, 18*(3–4), 258–265.

Percival, M. W., Wang, Y., Grosman, B., Dassau, E., Zisser, H., Jovanovic, L., et al. (2011). Development of a multi-parametric model predictive control algorithm for insulin delivery in type 1 diabetes mellitus using clinical parameters. *Journal of Process Control, 21*(3), 391–404.

Peyser, T., Dassau, E., Breton, M., & Skyler, J. S. (2014). The artificial pancreas: current status and future prospects in the management of diabetes. *Annals of the New York Academy of Sciences, 1311*, 102–123.

Rasmussen, C. E., & Williams, C. K. I. (2006). *Gaussian processes for machine learning*. MIT Press.
Ruan, Y., Thabit, H., Kumareswaran, K., & Hovorka, R. (2014). Pharmacokinetics of insulin lispro in type 2 diabetes during closed-loop insulin delivery. *Computer Methods and Programs in Biomedicine, 117*(2), 298–307.
Russell, S., & Norvig, P. (2009). *Artificial intelligence: A modern approach* (3.ª ed.). Prentice Hall.
Sanger, T. D. (2010). Controlling variability. *Journal of Motor Behavior, 42*(6), 401–407.
Sasi, A. Y. B., & Elmalki, M. A. (2013). Design and Analysis of a Sliding Table Controller for Diabetes. *Intelligent Control and Automation, 04*(03), 301–308.
Schölkopf, B., Mika, S., Burges, C. J. C., Knirsch, P., Muller, K.-R., Ratsch, G., et al. (1999). Input space versus feature space in kernel-based methods. *IEEE Transactions on Neural Networks, 10*(5), 1000–1017.
Schölkopf, B., & Smola, A. J. (2001). *Learning with kernels: support vector machines, regularization, optimization, and beyond* (1st ed.). The MIT Press.
Serhani, M. A., Benharref, A., & Nujum, A. R. (2014). An Adaptive Expert System for Automated Advices Generation-Based Semi-continuous M-Health Monitoring. In D. Ślęzak, A.-H. Tan, J. F. Peters, & L. Schwabe (Eds.), *Brain informatics and health* (pp. 388–399). Springer International Publishing.
Settles, B. (2010). Active learning literature survey.
Siegelaar, S. E., Holleman, F., Hoekstra, J. B. L., & DeVries, J. H. (2010). Glucose variability; does It matter? *Endocrine Reviews, 31*(2), 171–182.
Sigaud, O., Salaün, C., & Padois, V. (2011). On-line regression algorithms for learning mechanical models of robots: A survey. *Robotics and Autonomous Systems, 59*(12), 1115–1129.
Sorensen, J. T. (1985). *A physiologic model of glucose metabolism in man and its use to design and assess improved insulin therapies for diabetes* (Thesis). Massachussetts Institute of Technology.
Sutton, R. S., & Barto, A. G. (1998). *Reinforcement learning: An introduction*. MIT Press.
Turksoy, K., Bayrak, E. S., Quinn, L., Littlejohn, E., & Cinar, A. (2013). Multivariable Adaptive Closed-Loop Control of an Artificial Pancreas Without Meal and Activity Announcement. *Diabetes Technology & Therapeutics, 15*(5), 386–400.
Turksoy, K., & Cinar, A. (2014). Adaptive control of artificial pancreas systems – A review. *Journal of Healthcare Engineering, 5*(1), 1–22.
Ulas Acikgoz, S., & Diwekar, U. M. (2010). Blood glucose regulation with stochastic optimal control for insulin-dependent diabetic patients. *Chemical Engineering Science, 65*(3), 1227–1236.
Verdinelli, I., & Kadane, J. B. (1992). Bayesian designs for maximizing information and outcome. *Journal of the American Statistical Association, 87*(418), 510–515.
Weinzimer, S. A., Steil, G. M., Swan, K. L., Dziura, J., Kurtz, N., & Tamborlane, W. V. (2008). Fully automated closed-loop insulin delivery versus semiautomated hybrid control in pediatric patients with type 1 diabetes using an artificial pancreas. *Diabetes Care, 31*(5), 934–939.