# Approximate Control Design for Solar Driven Sensor Nodes

Clemens Moser[1], Lothar Thiele[1], Davide Brunelli[2] and Luca Benini[2]
[1] Swiss Federal Institute of Technology (ETH) Zurich
[2] University of Bologna

**Abstract.** This paper addresses power management of wireless sensor nodes which receive their energy from solar cells. In an outdoor environment, the future available energy is estimated and used as input to a receding horizon controller. We want to maximize the utility of the sensor application given the time-varying amount of solar energy. In order to avoid real-time optimization, we precompute offline an explicit state feedback solution. However, it is a well-known problem of the optimal feedback solution that the computational complexity grows very quickly, which is particularly unfavourable for sensor nodes. Given a standard sensor node platform and a typical, low-power sensing application, the storage and evaluation of an optimal feedback controller may introduce an unacceptable overhead. As a main contribution, we suggest a new approximation method to derive suboptimal control laws which substantially lower the computational and storage demand. We show that a sensor node's performance is not necessary decreased due to suboptimality of the control design. Hence our methods are potentially useful for resource-constrained systems like sensor nodes. All results are supported by simulations based on longterm measurements of solar energy in an outdoor setting.

## 1 Introduction

Wireless sensor networks (WSN) have opened up an exciting field of research that is increasingly becoming popular nowadays. A WSN can be seen as a system of self-powered, wireless sensors which are able to detect and transmit events to a base station. Main applications of WSNs are e.g. the monitoring of environmental physical quantities such as temperature, humidity or vibrations as well as physiological monitoring, smart spaces or factory instrumentation. Above all, sensor nodes are anticipated to be small and inexpensive devices which can be unobtrusively embedded in their environment [1]. Thus, a sensor node's hardware is stringently limited in terms of computation, memory, communication as well as storable energy (e.g. batteries). These resource constraints limit the complexity of the software executed on a sensor node.

Recently, techniques to harvest energy via photovoltaic cells have received increasing attention in the sensor network community [2]. In many applications, ambient solar energy can be used to recharge batteries and render frequent replacement of the batteries unnecessary. Ideally, sensor nodes once deployed benefit from a drastically increased operating time and become virtually immortal.

Two of the first prototype sensor nodes with energy harvesting capabilities were Heliomote [3] and Prometheus [4]. In both systems, the solar panels are directly connected with the storage device. In this way, the solar cell is operated in a manner that doesn't allow the cell to produce all the power it is capable of. An efficient solar harvesting system should adapt the electrical operating point of the solar cell to the given light condition, using techniques called Maximum Power Point Tracking (MPPT). For industrial, large-scale solar panels these techniques are well-understood and extensively used [5]. For solar cells the size of a few $cm^2$, however, particular care has to be taken in order not to waste the few $mW$ generated by the solar cell. The MPPT circuits proposed in [6, 7] are tailored to the needs of sensor nodes. That is, these circuits only pay off since their intrinsic power consumption is substantially lower than the amount of power they gain.

Clearly, the power generated by small solar cells is limited. Nodes executing a given application may frequently run out of energy in times with insufficient illumination. If one strives for predictable, continuous operation of a sensor node, common power management techniques have to be reconceived. In addition to perform classical power saving techniques, the sensor node has to adapt to the stochastic nature of solar energy. Goal of this adaptation is to maximize the utility of the application in a long-term perspective.

Concerning the software algorithms running on a sensor node, similar considerations as for the hardware hold. The energy required for sophisticated control algorithms may introduce a high control overhead for low-power applications. For sensor nodes which periodically sense and transmit data, but spend most of the time in power-saving sleep modes, simple, low-complexity solutions are needed.

A first step in this direction has been made in [8]. The authors point out how the problem of adapting the duty cycle of a solar powered sensor can be modelled by a linear program. As objective, the average duty cycle shall be optimized. Instead of periodically solving this linear program on-line, a heuristic algorithm of reduced complexity is proposed. The work in [9] improves on the results in [8]; however, the assumed optimization objective and application remain very specific. A more general approach to optimize the utilization of solar energy has been presented in [10]. In contrast to previous work, the class of linear programs presented is capable of modeling a much larger variety of application scenarios, constraints and optimization objectives. The basic idea is to apply multiparametric linear programming to obtain a piecewise linear state feedback over a polyhedral partition of the state space. In [11], this work is extended by introducing a hierarchical control approach which primarily improves the robustness of the system.

In multiparametric linear programming, the optimization problem is basically solved off-line and look-up tables are stored and evaluated in the on-line case. For state explosions, which occur already for problems of moderate complexity, the limited storage capabilities of sensor nodes are quickly exceeded. Furthermore, the evaluation of the numerous states will cost considerable time as well as energy. In this paper, a new algorithm for approximate multiparametric linear programming will be presented which generates much simpler look-up tables then the optimal solution. As far as we know, only one method has been proposed in the literature to get an approximate solution of a multiparametric linear programming problem [12]. Moreover, for a given look-up table, an efficient representation has been proposed which allows faster on-line evaluation [13].

The structure of the paper is as follows: In the next section, we outline our contributions. In Section 3, an overview of the system concept as well as the used models and methods are given. Section 4 reviews how optimal control laws are generated using multiparametric programming. In Sections 5, we present the key novelty of this paper, namely an algorithm for approximate multiparametric linear programming. Finally, experimental results are the topic of Section 6.

## 2 Contributions

In industrial applications, multiparametric solutions traditionally were not applied if processes with fast sampling rates had to be controlled. In this paper, we focus on a new field of application from the emerging area of wireless sensor networks. Here, it is mainly due to rigorous hardware and power constraints that complex control laws are prohibitive.

We propose a novel algorithm for approximate multiparametric linear programming. We demonstrate, that the online complexity of the generated control laws is highly reduced compared to an optimal solution in terms of computation overhead and storage demand. For an example application of practical relevance, we found improvements of 95% and 92%, respectively. For many applications, the optimal multiparametric solutions may grow to complex for constraint systems like sensor nodes. Moreover, with increasing complexity, well-established solvers come to their limits and fail to find the optimal solution. For all these applications, our algorithm may find useful approximations which exhibit performance metrics comparable to the optimal solution.

## 3 System Model

Fig. 1 illustrates the system model. The whole hardware/software system is powered by a photovoltaic module that delivers in a unit time interval starting at $t$ the energy $E_S(t)$. The sensor node may use the energy $E_S(t)$ directly to drive the application with energy $E_R(t)$. Surplus energy is stored in a storage device with efficiency $\eta$. At time $t$, there is the stored energy $E_C(t)$ available. The capability to bypass the storage device is a typical feature of latest prototypes [8]. It offers the opportunity to save substantial energy by using the solar energy directly when available.

Besides the application, there are two additional software tasks running on the target architecture. The estimator predicts future energy production of the harvesting device based on measurements of the past. The controller adapts properties of the application, e.g. task activation rates, based on the estimation

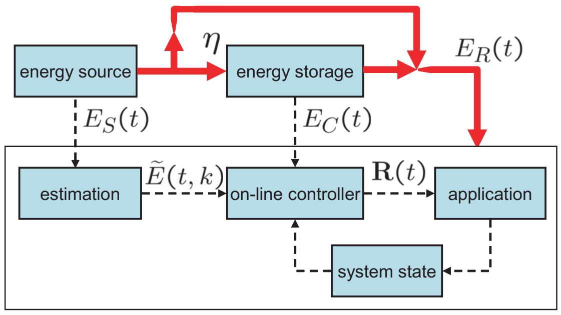


**Fig. 1.** Illustration of the system concept.

of future available energy, the currently stored energy and additional information about the system state, e.g. the amount of available data memory. Parameters of the application are modified by the on-line controller. During execution, the system state (e.g. the amount of information stored in local memory and the stored energy) is changed.

## 3.1 Power Flow

The modeling is based on the notion of discrete time $t \in \mathbb{Z}_{\geq 0}$ where the difference in physical time between two discrete time instances is denoted as $T$. Energy related sensing and control may happen only at times $t$. In a practical setting, one may have a basic time interval $T$ of a few minutes or even an hour.

The energy harvesting device is modeled as a power source which delivers energy $E_S(t)$ in the time interval $[t, t+1)$ of length $T$. Therefore, in time interval $[t_1, t_2)$ with $t_1, t_2 \in \mathbb{Z}_{\geq 0}$ it delivers energy $E_S(t_1, t_2) = \sum_{t_1 \leq u < t_2} E_S(u)$. The incoming power can be stored in an energy storage device, e.g. a rechargeable battery or a supercapacitor. In dependence on the energy $E_R(t)$ drawn from the sensor node, the increment $\Delta E_C(t)$ of the stored energy is defined at each time $t$ according to the following statement:

$$\begin{aligned} &\textbf{if}\, E_S(t) > E_R(t) \textbf{ then } \Delta E_C(t) = \eta \cdot (E_S(t) - E_R(t)) \\ &\qquad\qquad\qquad\qquad\quad \textbf{else } \ \Delta E_C(t) = (E_S(t) - E_R(t)) \end{aligned}$$

If the generated energy $E_S(t)$ is higher than energy $E_R(t)$, the sensor node is powered directly by the solar cell and excess energy is used to replenish the energy storage. As in [8], we account for the efficiency $\eta$ during the charging of the energy storage. As the arrangement in Fig. 1 is fully symmetrical, one could also consider $\eta$ when the storage is discharged. Alternatively, efficiencies of $\frac{\eta}{2}$ at both sides of the energy storage are thinkable; from a control point of view, all three mappings of the efficiency $\eta$ are equivalent.

The energy $E_R(t)$ is used to execute tasks on various system components. A task $\tau_i \in \mathbf{I}$ from the set of tasks $\mathbf{I}$ needs energy $e_i$ for completing a single instance. We suppose that a task is activated with a time-variant rate $r_i(t)$, i.e. during the basic time interval $T$ starting at $t$, the task is executed $r_i(t)$ times. Therefore, a task needs energy $E_i(t_1, t_2) = \sum_{t_1 \le u < t_2} e_i \cdot r_i(u)$ in time interval $[t_1, t_2)$ for successful execution. Finally, we denote $\mathbf{R}(t)$ the vector of all task rates $r_i$ at time $t$.

### 3.2 Receding horizon control

The estimation unit receives tuples $(t, E_S(t))$ for all times $t \ge 1$ and delivers $N$ predictions on the energy production of the energy source. We assume that the prediction intervals are of equal size denoted as the number $L$ (in units of the basic time interval $T$). We denote the total prediction horizon $H = N \cdot L$ (again in units of the basic time interval $T$), see also Fig. 2. At time $t$, the predictor produces estimations $\widetilde{E}_S(t + k \cdot L, t + (k+1) \cdot L)$ for all $0 \le k < N$. We write $\widetilde{E}(t, k) = \widetilde{E}_S(t + k \cdot L, t + (k+1) \cdot L)$ as a shorthand notation, i.e. the estimation of the incoming energy in the $(k+1)$st prediction interval after $t$.

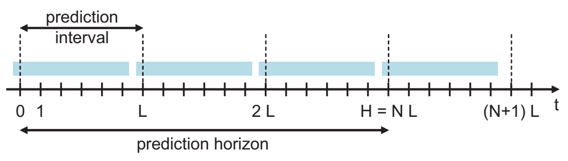


**Fig. 2.** Illustration of the prediction horizon.

At time $t$, the controller is computing the control sequence $\mathbf{R}(t + k \cdot L)$ for all prediction intervals $0 \le k < N$ based on the estimates $\widetilde{E}(t, k)$ as well as the current system state (e.g. $E_C(t)$). In other words, the rates of the different tasks $r_i$ are planned to be constant during each prediction interval. However, *only the first* control rates $\mathbf{R}(t)$ are applied to the system during the first time step $T$. The rest of the control sequence is discarded. At time $t + T$, a new vector $\mathbf{R}(t)$ is computed which extends the validity of the previous vector $\mathbf{R}(t-1)$ by one time step. Again only the first control is used, yielding a receding horizon control (RHC) strategy.

### 3.3 Linear program specification

The first step in constructing the on-line controller is the formulation of the optimization problem in form of a parameterized linear program (LP). In this paper, we restrict ourselves to the discussion of an example application which is modeled as a linear program (LP). However, for more sophisticated application models and more general linear program specifications, the reader is referred to [10].

Let us assume a sensor node is expected to observe some phenomenon of interest in an environmental monitoring application. For this purpose, an image has to be recorded with a camera sensor and the data has to be transmitted to a base station. We can model these requirements using a data sensing task $\tau_1$ and a data transmission task $\tau_2$. The data sensing task $\tau_1$ is operated with rate $r_1(t)$. At every instantiation, the sensing task $\tau_1$ drains $e_1$ energy units and stores an image in some local memory. The transmission task $\tau_2$ is transmitting images with a rate $r_2(t)$ and energy demand $e_2$. Thereby, task $\tau_2$ reduces the occupied memory by one image per instantiation. The corresponding linear program is given by (1).

$$\text{maximize } J = \mu \text{ subject to:} \qquad (1)$$

$$\begin{array}{cl}
r_1(t + j \cdot L) \geq \mu & \forall 0 \leq j < N \\
r_1(t + j), r_2(t + j) \geq 0 & \forall 0 \leq j < N \\
E_C(t + k \cdot L) = E_C(t) - \sum_{j=0}^{k-1} (e_1 \cdot r_1(t + j) + e_2 \cdot r_2(t + j)) + & \\
+ \sum_{j=0}^{k-1} \widetilde{E}(t, j) - (1 - \eta) \sum_{j=0}^{k-1} \lambda(j) & \forall 1 \leq k \leq N \\
\lambda(j) \geq \widetilde{E}(t, j) - e_1 \cdot r_1(t + j) - e_2 \cdot r_2(t + j) \geq 0 & \forall 0 \leq j < N \\
M(t + k) = M(t) + \sum_{j=0}^{k-1} (r_1(t + j) - r_2(t + j)) & \forall 0 \leq k < N \\
0 \leq M(t + j) \leq M_{max} & \forall 0 \leq j < N \\
E_C(t + N \cdot L) \geq E_C(t) &
\end{array}$$

The optimization objective of the linear program (1) is to maximize the minimal rate with which the task $\tau_1$ is operated in the finite horizon $0 \leq k < N$. In terms of intervals, the objective translates into a minimization of the maximum interval between any two consecutive measurements. This could be a reasonable objective if one attempts to minimize the unobserved time periods between two recorded images.

The auxiliary variable $\lambda$ accounts for the physical switching behaviour described in Section 3.1: In intervals when the energy provided by the source is higher then the energy demand of the sensor node, the energy storage is charged with efficiency $\eta$. The other way round, the energy storage is discharged in intervals when $\widetilde{E}(t, j)$ is low and $\lambda$ is forced to 0. The last inequality in LP(1) is a constraint on the energy at the end of the horizon which is needed to stabilize the receding horizon controller.

### 3.4 Hierarchical Control Design

It has been shown in [11] how linear programs like the one in (1) can be subdivided into two linear programs, each with its dedicated energy prediction algorithm (see Fig. 3). This reformulation is introduced here since it significantly improves the robustness of the system and it has been used in the experiments (see Sec. 6). Note, however, that a discussion on suitable energy prediction algorithms is beyond the scope of this paper.

The upper control layer is designed to avoid the depletion of the battery after a couple of cloudy days. Therefore, a worst-case energy prediction $\widetilde{E}_1(t, k)$

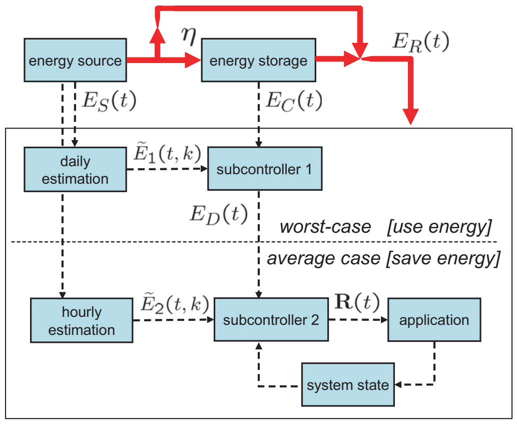


**Fig. 3.** Illustration of the hierarchical control model.

is used to predict the minimum available energy for the next days (prediction interval $L_1 \cdot T = 24h$, prediction horizon e.g. $H_1 \cdot T = 30days$). The objective of subcontroller 1 is to maximize the minimum available energy $E_D(t)$ per day.

maximize $J = \mu$ subject to: (subcontroller 1)

$$\begin{aligned} E_D(t+k\cdot L) &\geq \mu && \forall 0 \leq k < N \\ E_C(t+k\cdot L) &= E_C(t) + \widetilde{E}_1(t,k) - \textstyle\sum_{j=1}^{k} (E_D(t+(j-1)\cdot L)) && \forall 1 \leq k \leq N \\ 0 \leq E_C(t+k\cdot L) &\leq E_{max} && \forall 1 \leq k \leq N \end{aligned}$$

The energy $E_D(t)$ represents the maximum amount of energy which shall be used by subcontroller 2 at the next day. The sensing rate during the next day is set to $r_1 = \frac{\eta \cdot E_D(t)}{N_1 \cdot (e_1+e_2)}$. It is the duty of subcontroller 2 to adjust the rate $r_2$ of the energy hungry transmission task to optimally exploit the energy bypassing mechanism. To this end, an average energy prediction $\widetilde{E}_2(t,k)$ predicts the most likely energy values for a prediction horizon of $H_2 \cdot T = 24h$. For prediction intervals of a few hours (e.g. $L_2 \cdot T = 4h$), energy savings are maximized.

maximize $J = E_C(N)$ subject to: (subcontroller 2)

$$\begin{aligned} \textstyle\sum_{j=0}^{N_2-1} r_2(t+j) &= \frac{\eta \cdot E_D(t)}{e_1+e_2} && \\ r_2(t+j) &\geq 0 && \forall 0 \leq j < N \\ E_C(t+k\cdot L) &= E_C(t) - k\cdot e_1 \cdot r_1 - e_2 \cdot \textstyle\sum_{j=0}^{k-1} r_2(t+j) + && \\ &+ \textstyle\sum_{j=0}^{k-1} \widetilde{E}_2(t,j) - (1-\eta) \textstyle\sum_{j=0}^{k-1} \lambda(j) && \forall 1 \leq k \leq N \\ \lambda(j) \geq \widetilde{E}_2(t,j) - e_1 \cdot r_1 - e_2 \cdot r_2(t+j) &\geq 0 && \forall 0 \leq j < N \\ M(t+k) &= M(t) + \textstyle\sum_{j=0}^{k-1} (r_1(t+j) - r_2(t+j)) && \forall 0 \leq k < N \\ 0 \leq M(t+j) &\leq M_{max} && \forall 0 \leq j < N \end{aligned}$$

## 4 Multiparametric Linear Programming

Next, we will show how to design an on-line controller based on a state feedback law which avoids solving a linear program at each time step. Thereby, we are following the ideas in [14], where the regulation of discrete-time constrained linear systems is studied in the context of model predictive control (MPC). In [10], the application of multiparametric linear programming has been proposed for the first time for energy harvesting systems. We will briefly recall the main results.

As a first step, we define a state vector $\mathbf{X}$ consisting of the actual system state, the level of the energy storage as well as the estimation of the incoming energy over the finite prediction horizon (cp. Fig.1). For the linear program in (1), e.g., the state vector $\mathbf{X}$ can be written as

$$\mathbf{X}(t) = \Big( E_C(t)\,,\, M(t)\,,\, \widetilde{E}(t,0)\,,\, \ldots, \widetilde{E}(t, N-1) \Big)^T \tag{2}$$

Furthermore, let us denote the vector of optimal control inputs to the system, i.e., the vector of planned rates $\mathbf{R}$ as

$$\mathbf{U}^*(\mathbf{X}, t) = \left( \mathbf{R}^T(t)\,,\, \mathbf{R}^T(t+L)\,,\, \ldots, \mathbf{R}^T\,(t + (N-1)\cdot L)\,\right)^T. \tag{3}$$

The state space of $\mathbf{X}$ (in our case $\mathbb{R}^{N+2}$ bounded by possible constraints on $E_C(t), M(t)$ and $\widetilde{E}(t,i)$) can now be subdivided into a number $N_{CR}$ of polyhedrons. For each of these polyhedrons $i$ (also called critical regions) the optimal solution $\mathbf{U}^*(\mathbf{X})$ of the control problem can be made available explicitly as

$$\mathbf{U}^*(\mathbf{X}) = \mathbf{B}_j\mathbf{X} + \mathbf{C}_j \quad \text{if } \mathbf{H}_j\mathbf{X} \leq \mathbf{K}_j, j = 1, \ldots, N_{CR} \tag{4}$$

where $\mathbf{B}_j \in \mathbb{R}^{N\times(N+2)}, \mathbf{C}_j \in \mathbb{R}^N$ and $\mathbf{H}_j\mathbf{X} \leq \mathbf{K}_j, j = 1 \ldots N_{CR}$ is a polyhedral partition of the state space of $\mathbf{X}$. For simplicity, we dropped the dependence on $t$ of the state vector $\mathbf{X}$. The computation of the vectors and matrices of control law (4) is done off-line using, e.g., the algorithm presented in [15] or other efficient solvers cited in the latter work.

In the on-line case, the controller has to identify to which region $j$ the current state vector $\mathbf{X}$ belongs. After this membership test, the optimal control moves $\mathbf{U}^*$ for the next $N$ prediction intervals may be computed by evaluating a linear function of $\mathbf{X}$. However, according to the receding horizon policy it is sufficient to calculate only the first rates $\mathbf{R}(t)$ for the next interval. These rates $\mathbf{R}(t)$ are identical to the rates one would obtain by solving the linear program. However, the computational demand is greatly reduced compared to solving a LP on-line. After having solved the mp-LP in advance, a set of $N_{CR}$ polyhedra with associated control laws has to be stored and evaluated at each time step $t$. The computation demand in the online case now depends on

- the number of critical regions $N_{CR}$ which have to be tested,

- the size of the state vector $\mathbf{X}$ (in particular the number of prediction intervals $N$),
- and finally on the number of controlled rates $\mathbf{R}$ which have to be determined.

If the number of critical regions $N_{CR}$ gets large, the computational effort still may be large as many tests of the form $\mathbf{H}_j\mathbf{X} \leq \mathbf{K}_j$ must be performed. Typically, the computational effort spent for these matrixmultiplications is much higher then evaluating the linear function $\mathbf{B}_j\mathbf{X} + \mathbf{C}_j$. In related work, there have been proposals how the on-line complexity of a given control law (4) can be reduced. In [13], this issue has been addressed by finding a representation of the polyhedral partition which allows more efficient region testing. Nevertheless, those techniques come to their end if the number $N_{CR}$ of critical regions is high. Indeed, this general shortcoming of the mp-LP approach may render the calculated controllers inapplicable for resource constrained sensors. By proposing approximate, sub-optimal feedback controllers we will show how complex control problems can be mastered anyhow. To this end, we try to reduce the number of critical regions $N_{CR}$ for a given control problem.

## 5 Approximative MP Linear Programming

It is well known that the size of the explicit solution obtained by multiparametric linear programming grows quickly if the complexity of the control problem increases. In deed, already a moderate number of control variables and parameters may result in a huge number $N_{CR}$ of critical regions. At this, adjacent regions are often characterized by almost identical control laws. This circumstance, however, has in many cases neglectable impact on the resulting control profile. Rather, numerous regions $N_{CR}$ entail an high overhead in terms of storage requirement, running time as well as energy consumption.

Beyond these general short-comings, one could also argue in favour of an approximate control approach due to the stochastic nature of the harvested energy. The future energy provided by photovoltaic cells can only be estimated. The predicted energy values, which are available in practice may be too inexact and too unreliable to provide the basis for "exact" procedures like MP linear programming. From this point of view, a precise calculation of the optimal control values turns out to be worthless if major prediction errors occur.

In this section, we present a new algorithm for approximative multiparametric linear programming. The basic idea is

- to take a large number of samples $\mathbf{X}_i$ of the state space of $\mathbf{X}$ (compare equation (2)),
- to solve a linear program for each sample $\mathbf{X}_i$ to obtain the respective optimal solution $\mathbf{U}_i^*$,
- to find a (preferably simple) fitting function $\hat{\mathbf{U}}^*(\mathbf{X})$ for the multidimensional data $(\mathbf{X}_i, \mathbf{U}_i^*)$,
- and finally to use $\hat{\mathbf{U}}^*(\mathbf{X})$ (which has been calculated offline) as approximation for $\mathbf{U}^*(\mathbf{X})$ in the online case.

At first, a random number generator is used to generate the samples $\mathbf{X}_i$, $1 \leq i \leq N_S$, where $N_S$ denotes the total number of samples. We used independent, uniformly distributed random values as samples for the single elements of $\mathbf{X}$. For example, values of the stored energy $E_C$ have been chosen according to a uniform distribution

$$f_{E_C}(E_C) = \begin{cases} \frac{1}{E_{max}} & \text{if } 0 < E_C < E_{max} \\ 0 & \text{else} \end{cases}, \tag{5}$$

with the probability density function $f_{E_C}(E_C)$ and the maximum storable energy $E_{max}$. In the same way, we sampled $M$ and $\widetilde{E}$ using the respective upper bounds on the available memory as well as producible energy.

As fitting algorithm, we opted for the algorithm proposed in [16]. This algorithm attempts to fit data samples to a set of convex, piece-wise linear candidate functions. In order to provide a good fit, the algorithm requires the function which generates the samples to have a convex curvature. The optimal control rates $\mathbf{U}^*(\mathbf{X})$, however, are not necessary convex over the state space $\mathbf{X}$. Hence, a direct fitting of the control rates is not possible using the algorithm in [16].

It has been shown that the optimal objective value $J^*(\mathbf{X})$ exhibits the wished convexity property.

**Theorem 1 (cf. page 180 in [17]).** *The function $J^*(\mathbf{X})$ is continuous, piece-wise affine and convex over $\mathbf{X}$.*

As the optimal control vector $\mathbf{U}^*(\mathbf{X})$, $J^*(\mathbf{X})$ can be computed exactly using multiparametric programming as

$$J^*(\mathbf{X}) = \mathbf{T}_j\mathbf{X} + \mathbf{V}_j \quad \text{if } \mathbf{H}_j\mathbf{X} \leq \mathbf{K}_j, j = 1, \ldots, N_{CR} \tag{6}$$

where $\mathbf{T}_j$ and $\mathbf{V}_j$ are matrices of appropriate dimensions. It is important to note that $J^*(\mathbf{X})$ is piecewise linear over the *same* polyhedral partition $\mathbf{H}_j\mathbf{X} \leq \mathbf{K}_j$ as the optimal control $\mathbf{U}^*(\mathbf{X})$ (cf. equation (4)).

For each sample $\mathbf{X}_i$, we now solve a linear program and determine the optimal control vector $\mathbf{U}^*(\mathbf{X}_i)$ as well as the optimal objective value $J^*(\mathbf{X}_i)$. This can be done using common simplex-based or interior-point solvers. Next, we implement the heuristic algorithm in [16] to fit the objective $J^*(\mathbf{X_i})$, i.e to solve the least square fitting problem

$$\text{minimize} \quad \sum_{i=1}^{N_S} \left( \max_{j=1,\ldots,\hat{N}_{CR}} (\hat{\mathbf{T}}_j^T \cdot \mathbf{X}_i + \hat{\mathbf{V}}_j) - J^*(\mathbf{X}_i) \right)^2 \tag{7}$$

Like that, we obtain the approximated objective function $\hat{J}^*(\mathbf{X})$ in the so-called "max-affine" form:

$$\hat{J}^*(\mathbf{X}) = \max_{j=1,\ldots,\hat{N}_{CR}} \{\hat{\mathbf{T}}_j^T \cdot \mathbf{X} + \hat{\mathbf{V}}_j\} \tag{8}$$

The piecewise affine convex function (8) can be recast easily in the following equivalent form which explicitly defines the polyhedral partition $\hat{\mathbf{H}}_j\mathbf{X} \leq \hat{\mathbf{K}}_j$ (see also [18]).

$$\hat{J}^*(\mathbf{X}) = \hat{\mathbf{T}}_j\mathbf{X} + \hat{\mathbf{V}}_j \quad \text{if } \hat{\mathbf{H}}_j\mathbf{X} \leq \hat{\mathbf{K}}_j, j = 1, \ldots, \hat{N}_{CR} \tag{9}$$

Next, we group the samples $\mathbf{X}_i$ according to the region $j$ they belong to. For each region $j$, we perform a simple least square fitting of the respective samples to compute the coefficients $\hat{\mathbf{A}}_j$ and $\hat{\mathbf{B}}_j$ of the approximated control rates $\hat{\mathbf{U}}^*$. As a result, we have derived an explicit form for the control rates $\hat{\mathbf{U}}^*(\mathbf{X})$ as a function of the current state $\mathbf{X}$:

$$\hat{\mathbf{U}}^*(\mathbf{X}) = \hat{\mathbf{A}}_j\mathbf{X} + \hat{\mathbf{B}}_j \quad \text{if } \hat{\mathbf{H}}_j\mathbf{X} \leq \hat{\mathbf{K}}_j, j = 1, \ldots, \hat{N}_{CR} \tag{10}$$

Everything done so far has to be done off-line. The approximated control law in (10) can now be used in an on-line controller instead of the exact solution in (4). Since the convex fitting algorithm in [16] allows to tune the number $\hat{N}_{CR}$ of critical regions, one may chose a smaller number of regions $\hat{N}_{CR} < N_{CR}$ to reduce the complexity of the control problem. The fitting algorithm then attempts to create a smaller polyhedral partition which minimizes the least square error of the objective value function. In comparison to the optimal, multiparametric solution the fitting algorithm merges smaller regions and reshapes the geometry of the partition, as we will see in the next section. Albeit no performance guarantees of the heuristic algorithm are given in [16], it turns out that the algorithm performs well and produces suitable approximations, both in [16] and also in our experiments.

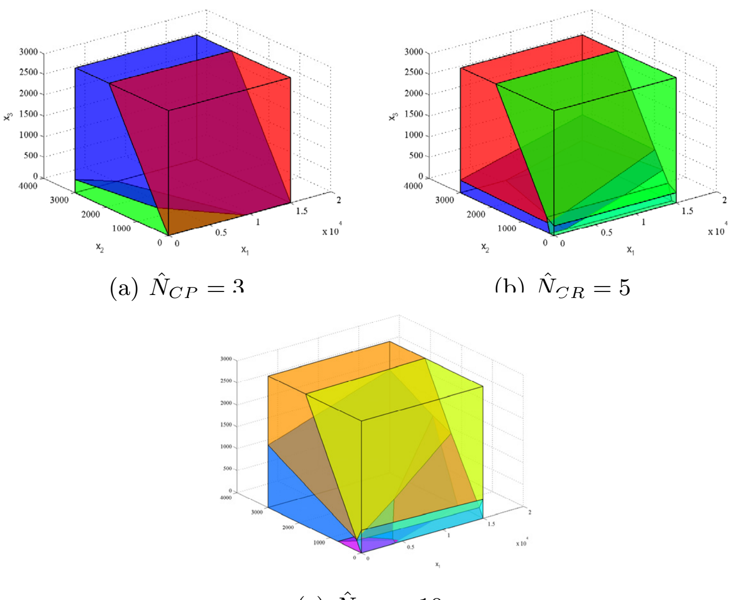
It is worth noting that for an efficient on-line implementation as in [13], we did not implement the controller given in (10) directly. Rather, we used the "max-affine" representation of the approximated objective function $\hat{J}^*(\mathbf{X})$ in (8) to identify the currently active region $j$. In this way, far less multiplications, additions and comparisons are required to find the maximum term $\{\hat{\mathbf{T}}_j^T \cdot \mathbf{X} + \hat{\mathbf{V}}_j\}$ than to evaluate the terms $\hat{\mathbf{H}}_j\mathbf{X} \leq \hat{\mathbf{K}}_j$ in (10). In addition, the storage demand is significantly reduced since it is not necessary to store the polyhedral regions anymore. Finally, the control rates $\hat{\mathbf{U}}^*(\mathbf{X}) = \hat{\mathbf{A}}_j\mathbf{X} + \hat{\mathbf{B}}_j$ are evaluated for the identified region.

## 6 Experimental Results

We implemented the multiparametric controllers for an exemplary case study using the MATLAB toolbox in [19]. Based on experimental results with our own prototype energy scavenger, we opted for a storage efficiency of $\eta = 80\%$. On the other hand, measurements of solar light intensity during nearly 5 years recorded at [20] serve as energy input $E_S(t)$. The time interval between two samples is 5 minutes, so we set the basic time interval $T = 5\,\text{min}$. Using this data, we could extensively test the performance of our algorithms for time scales one usually wants to achieve with solar powered sensor networks.

The key operation in finding an approximate solution for a control problem is the fitting of the (convex) objective function $\hat{J}^*(\mathbf{X}) = \max_{j=1,\ldots,\hat{N}_{CR}} \{\hat{\mathbf{T}}_j^T \cdot \mathbf{X} + \hat{\mathbf{V}}_j\}$. At this, the algorithm in [16] alternates between partitioning the data in new regions $j$ and carrying out least-squares fits to update the coefficients $\hat{\mathbf{T}}_j$ and $\hat{\mathbf{V}}_j$. Starting with an initial number $\hat{N}_{CR,init}$, critical regions may be merged at every iteration, leading to a reduced number of regions. The algorithm converges if a partition remains unchanged after an iteration or some maximum number of iterations is reached. The final number of partitions $\hat{N}_{CR}$ can be influenced to some extent by appropriate choice of the initial parameters.

For subcontroller 1, the state space $\mathbf{X}_1 = \left(E_C(t), \widetilde{E}_1(t,0), \ldots, \widetilde{E}_1(t,29)\right)$ was sampled using $N_S = 1000$ random samples. The calculated control laws devide this state space in $\hat{N}_{CR} \leq 6$ critical regions. In other words, with the described method we could not generate more than 6 partitions. For subcontroller 2, $N_S = 2000$ samples have been taken from the state space $\mathbf{X}_2 = \left(E_D(t), M(t), \widetilde{E}_2(t,0), \ldots, \widetilde{E}_2(t,5)\right)$. Here, the approximation algorithm converges towards $\hat{N}_{CR} \leq 10$ critical regions. In Fig. 4, some exemplary partitions of $\mathbf{X}_2$ are displayed. In dependence of the choice of the 3-dimensional cut, not all $\hat{N}_{CR}$ partitions may be visible in diagrams (a)-(c).



(a) $\hat{N}_{CP} = 3$

(b) $\hat{N}_{CR} = 5$

(c) $\hat{N}_{CR} = 10$

**Fig. 4.** Illustration of approximated polyhedral partitions for subcontroller 2. Cut through $\widetilde{E}_2(t,1) = 1200$, $\widetilde{E}_2(t,2) = 1000$, $\widetilde{E}_2(t,3) = 0$, $\widetilde{E}_2(t,4) = 100$, $\widetilde{E}_2(t,5) = 500$.

Let us resume the hierarchical control model for the camera application which has been introduced in Section 3.4. Generally, radio communication is the main energy consumer an a sensor node. Hence, we choose $e_1 = 0.1$ and $e_2 = 0.9$ as energy demands for the sensing and the transmission task, respectively. The maximum storage capacity of the sensor node is $M_{max} = 1000$. To avoid unnecessary control overhead, subcontroller 1 is not activated every $T = 5$ minutes, but only once per day.

We denote $\hat{r}_1$, $\hat{r}_2$, $\hat{E}_C$ and $\hat{M}$ the rate and state variables obtained if approximate control laws are applied. Let us also define the rectifier function $[\varDelta E]^+$ as follows:

$$[\varDelta E]^+ = \begin{cases} \varDelta E & \text{if } \varDelta E \geq 0 \\ 0 & \text{if } \varDelta E < 0 \end{cases} \tag{24}$$

Using this notation, we denote the average efficiency of the energy utilisation

$$\eta_{avg} = 1 - \frac{(1-\eta) \cdot \sum\limits_t [E_S(t) - e_1 r_1(t) - e_2 r_2(t)]^+}{\sum\limits_t E_S(t)} \tag{25}$$

In the following, the efficiency $\eta_{avg}$ will be used as a metric to quantify the performance of subcontroller 2.

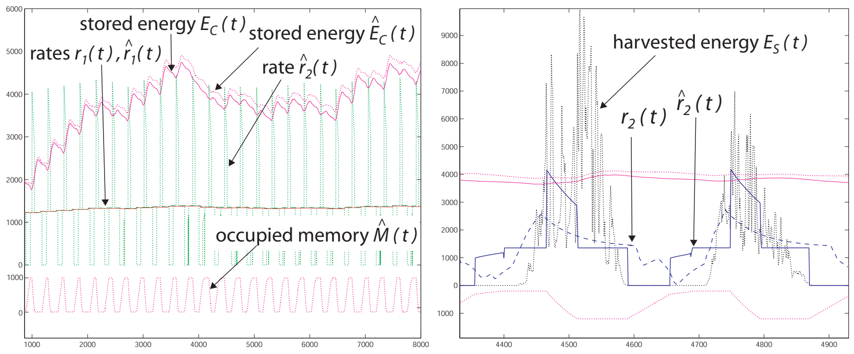


**Fig. 5.** Hierarchical system, approximate vs. optimal multiparametric programming, $\hat{N}_{CR} = 4$ for both subcontroller 1 and subcontroller 2.

Fig. 5 displays the evaluation of an approximate control law with $\hat{N}_{CR} = 4$ for both subcontroller 1 and subcontroller 2. In comparison, the optimal solution exhibits 30 and 161 critical regions, respectively (see Table 1). The primary optimization objective of regulating the sensing rate $r_1$ is met almost as well as the exact solution. The maximal derivation of $\hat{r}_1$ from $r_1$ is 1.52%, making both lines indistinguishable in the left diagram of Fig. 5. For both control designs, the rate $r_1$ is optimized in spite of the unstable power supply $E_S(t)$. Consequently the, stored energy $E_C(t)$ is increasing during day and decreasing at night.

Also the transmission rate $r_2$ is oscillating around the sampling rate $r_1$. Since it is favourable to use energy when available, data is stored at night and transmitted during day. Obviously, the approximated algorithm manages to save even slightly more energy then its exact counterpart. As displayed in the right diagram of Fig. 5, the transmission rate $\hat{r}_2$ is adjusted to much higher values during day and consequently set to 0 at night. Apparently, this strategy even yields a gain of 0.75% of the average efficiency $\eta_{avg}$. The stored energy $\hat{E}_C$ is varying up to 11.57% from $E_C$. However, the peak of $\hat{E}_C$ is just 4.03% above the one of $E_C$. That is, the capacity of the energy storage is required to be approximately 5% higher if the system is controlled by an approximated algorithm.

Showing a comparable performance during runtime, the main advantage of the approximation becomes obvious considering the complexity of the control laws. According to Table 1, the storage demand is significantly reduced by 92.44% compared to the optimal solution. In terms of worst case computation demand, the reduction even amounts 95.57%. Here, worst case refers to the situation where the currently active region is the last region to be tested (cp. Equation 4). Table 1 also outlines the results for a second low-complexity approximation. It exhibits a slightly lower efficiency $\eta_{avg}$. On the other hand the second approximation closely matches the optimal case in terms of control rate $r_1$.

**Table 1.** Comparison of multiparametric and approximate-mp control design, sc = subcontroller, storage in real numbers, ops in the worst case.

| control design | $\max_t \left\| \frac{\hat{r}_1(t)}{r_1(t)} - 1 \right\|$ | $\max_t \left\| \frac{\hat{E}_C(t)}{E_C(t)} - 1 \right\|$ | $\eta_{avg}$ | $N_{CR}$ (or $\hat{N}_{CR}$) | storage | ops |
|---|---|---|---|---|---|---|
| optimal, sc1 | 0% | 0% | 93.00% | 30 | 1920 | 3689 |
| sc2 | | | | 161 | 2898 | 4829 |
| approximate, sc1 | 1.52% | 11.57% | 93.75% | 4 | 256 | 308 |
| sc2 | | | | 4 | 108 | 69 |
| approximate, sc1 | 0.82% | 5.47% | 92.97% | 4 | 256 | 308 |
| sc2 | | | | 9 | 243 | 173 |

The dimensions of the approximated control laws are now of the same order as those analyzed and implemented in [10]. In the latter work, running times as well as power consumptions of corresponding algorithms have been measured on a BTnode [21]. Therefore we conclude that the algorithms derived in this section can be implemented efficiently on a real sensor node, involving neglectable implementation overhead.

In summary, we can state that the approximate solutions to the multiparametric control problem do not necessarily entail a degraded performance in terms of the controlled parameters. In deed, in most cases we could find simple but useful approximations for the underlying control problem. Besides the significant complexity reduction, there is another advantage of the proposed approximation technique that should be mentioned. For highly complex control problems consisting of several thousands of regions $N_{CR}$, conventional solvers may be unable to find the optimal polyhedral partition. We experienced that sometimes even no solution can be found at all. In other examples, the mp-LP solvers we used could not generate the optimal partition with the minimal number of regions $N_{CR}$. Rather, a high number of overlapping regions is constructed which cannot be removed afterwards. Here, our method turned out to be helpful to find a reasonable solution at all.

## 7 Conclusion

In this paper, we present a specification model that is suited to capture the performance, the parameters and the energy model of solar powered sensor nodes. This challenging field of application is characterized by strict hardware and software constraints. We propose a new algorithm for approximative multiparametric linear programming. The resulting control laws are rough approximations of the optimal solution and reduce the involved online overhead substantially. An experimental setup reveals that the achieved performance of selected control laws may be comparable to the optimal solution. All methods are supported by extensive simulations results which are based on long-term measurements of solar energy.

## Acknowledgements

The work presented in this paper was partially supported by the National Competence Center in Research on Mobile Information and Communication Systems (NCCR-MICS), a center supported by the Swiss National Science Foundation under grant number 5005-67322. In addition, this research has been founded by the European Network of Excellence ARTIST2.

## References

1. K. Roemer and F. Mattern, "The design space of wireless sensor networks," in *IEEE Wireless Communications*, vol. 11, no. 6, Dec. 2004, pp. 54–61.
2. V. Raghunathan, A. Kansal, J. Hsu, J. Friedman, and M. B. Srivastava, "Design considerations for solar energy harvesting wireless embedded systems." in *Proceedings of the Fourth International Symposium on Information Processing in Sensor Networks, IPSN 2005*, UCLA, Los Angeles, California, USA, April 25-27 2005, pp. 457–462.
3. J. Hsu, A. Kansal, J. Friedman, V. Raghunathan, and M. Srivastava, "Energy harvesting support for sensor networks." in *SPOTS track at IPSN 2005*, 2005.

4. X. Jiang, J. Polastre, and D. E. Culler, “Perpetual environmentally powered sensor networks.” in *Proceedings of the Fourth International Symposium on Information Processing in Sensor Networks, IPSN 2005*, UCLA, Los Angeles, California, USA, April 25-27 2005, pp. 463–468.
5. T. Esram and P. Chapman, “Comparison of photovoltaic array maximum power point tracking techniques,” *Energy Conversion, IEEE Transaction on*, vol. 2, pp. 439–339, 2007.
6. F. Simjee and P. H. Chou, “Everlast: long-life, supercapacitor-operated wireless sensor node,” in *ISLPED ’06: Proceedings of the 2006 international symposium on Low power electronics and design.* New York, NY, USA: ACM Press, 2006, pp. 197–202.
7. C. Park and P. Chou, “Ambimax: Autonomous energy harvesting platform for multi-supply wireless sensor nodes,” in *Proceedings of the Sensor and Ad Hoc Communications and Networks. SECON ’06.*, vol. 1, 2006, pp. 168–177.
8. A. Kansal, J. Hsu, S. Zahedi, and M. B. Srivastava, “Power management in energy harvesting sensor networks,” *Trans. on Embedded Computing Sys.*, vol. 6, no. 4, p. 32, 2007.
9. C. M. Vigorito, D. Ganesan, and A. G. Barto, “Adaptive control of duty cycling in energy-harvesting wireless sensor networks,” in *4th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON 2007)*, 2007, pp. 21–30.
10. C. Moser, L. Thiele, D. Brunelli, and L. Benini, “Adaptive Power Management for Energy Harvesting Systems,” in *Design, Automation and Test in Europe (DATE 07)*, Nice, France, April 16-20 2007.
11. ——, “Robust and Low Complexity Rate Control for Solar Powered Sensors,” in *submitted to Design, Automation and Test in Europe (DATE 08).*
12. C. Filippi, “An algorithm for approximate multiparametric linear programming,” *Journal of Optimization Theory and Applications*, vol. 120, no. 1, pp. 73–95(23), January 2004.
13. F. Borrelli, M. Baotic, A. Bemporad, and M. Morari, “Efficient On-Line Computation of Constrained Optimal Control,” in *IEEE Conference on Decision and Control*, Orlando, Florida, Dec. 2001, pp. 1187–1192. [Online]. Available: http://control.ee.ethz.ch/index.cgi?page=publications;action=details;id=287
14. A. Bemporad, F. Borrelli, and M. Morari, “Model Predictive Control Based on Linear Programming - The Explicit Solution,” *IEEE Transactions on Automatic Control*, vol. 47, no. 12, pp. 1974–1985, Dec. 2002. [Online]. Available: http://control.ee.ethz.ch/index.cgi?page=publications;action=details;id=299
15. F. Borrelli, A. Bemporad, and M. Morari, “A Geometric Algorithm for Multi-Parametric Linear Programming,” *Journal of Optimization Theory and Applications*, vol. 118, no. 3, pp. 515–540, Sept. 2003. [Online]. Available: http://control.ee.ethz.ch/index.cgi?page=publications;action=details;id=236
16. A. Magnani and S. Boyd, “Convex piecewise-linear fitting.” in *Optimization and Engineering (submitted)*, http://www.stanford.edu/~boyd/reports/cvx_pwl_fit.pdf, April 2006.
17. T. Gal, “Postoptimal Analyzes, Parametric Programming, and Related Topics,” 2nd ed., Berlin, Germany,de Gruyter 1995.
18. M. Schechter, “Polyhedral functions and multiparametric linear programming,” *Journal of Optimization Theory and Applications*, vol. 53, no. 2, pp. 269–280, 1987.
19. M. Kvasnica, P. Grieder, and M. Baotić, “Multi-Parametric Toolbox (MPT),” 2004. [Online]. Available: http://control.ee.ethz.ch/~mpt/

20. Bern University of Applied Sciences, Engineering and Information Technologies, Photovoltaic Laboratory, "Recordings of solar light intensity at Mont Soleil from 01/01/2002 to 31/09/2006," www.pvtest.ch, March, 2007.
21. J. Beutel, M. Dyer, M. Hinz, L. Meier, and M. Ringwald, "Next-generation prototyping of sensor networks," in *Proc. 2nd ACM Conf. Embedded Networked Sensor Systems (SenSys 2004).* ACM Press, New York, Nov. 2004, pp. 291–292.