Proceedings of the
44th IEEE Conference on Decision and Control, and
the European Control Conference 2005
Seville, Spain, December 12-15, 2005

On approximate policy iteration for continuous-time systems

Andreas Wernrud and Anders Rantzer
Department of Automatic Control
Lund Institute of Technology
Box 118, SE-221 00 Lund, Sweden
{andreas, rantzer}@control.lth.se

Abstract— **We propose a new algorithm for feedback nonlinear synthesis. The algorithm computes suboptimal solutions, with bounds on suboptimality, to the Hamilton-Jacobi-Bellman equation. For systems that are modeled with polynomials the computations can be done efficiently via semidefinite programming. To illustrate the strength of the proposed method, we compute smooth stabilizing feedback controllers for several problems.**

I. INTRODUCTION

Control synthesis for nonlinear systems is well known to be difficult. Many methods have been proposed, some of them have also been proved to work well for specific problems. Still, no general method exists. To guide the search for controllers which result in good performance the synthesis problem can be posed in the framework of optimal control. In this way we can obtain a characterization of the problem in terms of the Hamilton-Jacobi-Bellman(HJB)-equation. Consider for example the case with linear dynamics and quadratic cost function. In this case the HJB-equation reduces to a Riccati equation. Since Riccati equations can be solved efficiently, optimal control of linear systems with quadratic cost, LQ-control, is a practical design tool.

On the other hand, if the assumption on linear dynamics or quadratic cost does not hold, the resulting HJB-equation is very difficult to solve. Indeed there does not exist any closed form solution, except for very special problems. Even more severe, there does not exist a general computational scheme for obtaining approximate solutions. Due to its importance much time have been devoted to find such schemes.

In [5], [10] the authors used various power series expansion strategies, with various assumptions, to obtain approximate solutions to the HJB-equation. These methods can sometimes be used to compute good local estimates, using only a few terms. Although higher order approximations are possible to compute it is often too difficult. Also, the region where the approximations are valid are not well defined.

Motivated by LQ-control, another approach is to write the nonlinear system in a linear like representation and to derive a state dependent Riccati equation, see [7]. A particular such approach is taken in [16], where the authors use representations of positive polynomials to derive sufficient conditions for upper bounds on the value function.

In [9] the authors uses discretization-interpolation techniques. The state space is discretized and the open loop minimum control is computed for each point. These are then combined to form a feedback controller. The drawback is that gridding techniques are expensive in that such methods require computations that scale exponentially in state dimension.

In this paper we take a different approach, namely policy iteration or successive approximation in policy space. Related methods, also successive approximation methods, was already discussed by Bellman in [3]. We will follow the work presented in [8]. In that paper the first theoretical analysis of policy iteration is given. They show how the solution of the HJB-equation can be reduced to a sequence of first order linear partial differential equations. They derive useful inequalities and prove convergence of the algorithm. However, the work is mostly of theoretical nature and they do not propose a computational procedure. Such a procedure was proposed in [1]. In that work the Galerkin spectral method is used to obtain approximate solutions to the aforementioned sequence of linear partial differential equations. However, no bounds on the approximations are given, initial iterates must be provided and assumptions which are difficult to check must be fulfilled. Moreover, the main computational task in the algorithm proposed in [1] is multidimensional integration. Such computations become prohibitive for systems with more then a few states. On the other hand, an advantage of that method is that it can handle problems where the system dynamics are not necessarily modeled with polynomials. This is in contrast to the method presented in this paper.

This paper introduces a new version of policy iteration. The main idea is to replace an equality constraint with two inequalities. With this modification we can apply convex optimization to solve the intermediate steps in the algorithm. We also get bounds on how far from optimality each iterate is.

II. DEFINITIONS AND KNOWN FACTS

A. Problem statement

We start with a problem description. Consider a nonlinear dynamical system

$$\dot{x}(t) = f(x(t)) + g(x(t))u(t) \tag{1}$$

with $(x(t), u(t)) \in X \times U \subset (\mathbb{R}^n \times \mathbb{R}^m)$ and $(0,0) \in X \times U$. To start with, we assume that $f, g \in C(X, \mathbb{R}^n)$. We shall consider regulation at the origin, thus we require that $f(0) = 0$. Let $l : \mathbb{R}^n \to \mathbb{R}$ be continuous and positive for $x \neq 0$ and $l(0) =$

0, also let $R > 0$ be positive definite matrix. We define the instantaneous cost to be

$$q(x,u) = l(x) + |u|_R^2 \text{ on } X \times U$$

Define the admissible control set as $U(X) = C(X,U)$. As a measure of system performance under input u we will consider the cost function

$$V(x_0,u) = \int_0^\infty q(\phi(t,x_0,u), u(\phi(t,x_0,u)))dt \tag{2}$$

where $\phi : \mathbb{R} \times X \times U \to X$ is the trajectory for (1), when starting from x_0 and using input u. To simplify notation, we will implicitly require all cost functions, or approximations of such, to vanish at the origin and to be positive for $x \in X \setminus \{0\}$. The problem we will address is the computation of

$$V^*(x_0) = \inf_{u \in U(X)} V(x_0,u) \quad \forall x_0 \in X \tag{3}$$

The function V^* is called the optimal value function. The following result is well known,

Proposition 1: (Hamilton-Jacobi-Bellman-equation) Let $V^* \in C^1(X,\mathbb{R})$ and $u^* \in U(X)$ be such that

$$\min_u \{ \frac{\partial V^*(x)}{\partial x}^T (f(x) + g(x)u) + l(x) + |u|_R^2 \} = 0, \quad \forall x \in X \tag{4}$$

and that the minimum is attained at $u^* \in U(X)$. Then $\forall x_0 \in X$ the trajectory $\phi(t,x_0,u^*)$ exists $\forall t \geq 0$. Moreover, $V^*(x_0,u^*) = V(x_0,u^*)$ and the minimizing controller is given by

$$\begin{aligned} u^*(x) =& \arg\min_u \{ \frac{\partial V^*(x)}{\partial x}^T (f(x) + g(x)u) + l(x) + |u|_R^2 \} \\ =& -\frac{1}{2} R^{-1} \frac{\partial V^*(x)}{\partial x}^T g(x), \quad \forall x \in X \end{aligned} \tag{5}$$

In most cases V^* is not a differentiable function. Therefore the solution to (4) must be interpreted in generalized sense. The machinery of viscosity solutions resolves issues of existence and uniqueness. We will not discuss this further in this paper, see [4]. It is important to note that the methods developed in this paper do not require V^* to be differentiable. Substituting u^* back in (4) gives

$$l(x) + \frac{\partial V^*(x)}{\partial x}^T f(x) - \frac{1}{4} \frac{\partial V^*(x)}{\partial x}^T g(x) R^{-1} g(x)^T \frac{\partial V^*(x)}{\partial x} = 0 \tag{6}$$

Now the original problem (3) has been reduced to the solution of this nonlinear partial differential equation for V^*. Although this is a simplification, the solution of (6) is still very difficult and no general closed form solution exists. In the next section we describe how the solution can be obtained from the solution of a sequence of simpler problems.

B. Exact policy iteration

Suppose that the cost function (2) is finite on X when using input u. If we differentiate along the corresponding trajectory we obtain

$$\frac{\partial V(x)}{\partial x}^T (f(x) + g(x)u(x)) + l(x) + |u(x)|_R^2 = 0 \tag{7}$$

which is an equation for the instantaneous cost-decrease. The equation does not involve the solution trajectory so it should be easier to obtain an expression for V via this equation, compared to solving for the system trajectory and compute (2). To simplify notation we define H by

$$H(V) = u = -\frac{1}{2} R^{-1} \frac{\partial V(x)}{\partial x}^T g(x), \quad V \in C^1(X,\mathbb{R})$$

Also, for given $u \in U(X)$ we denote

$$V = T(u) \tag{8}$$

as the solution of equation (7). These observations, together with the next proposition, suggests a computational procedure.

Proposition 2: Let $u \in U(X) \bigcap \{u : V(x_0,u) < \infty\}$, if there exists a solution $V \in C^1(X,\mathbb{R}) \bigcap \{V : V(0) = 0\}$ to $V = T(u)$ then V is unique and $V(x) = V(x,u(x))$. If $u_+ = H(V)$ then, $V(x,u(x)) \geq V_+(x,u_+(x))$, where $V_+ = T(u_+)$. If equality holds we must have $V^* = V$.

The proof is essentially contained in [8]. Thus, under the assumption that we can find the exact solution to (8) for a given u, the conclusion of the proposition is that we can obtain a monotonically converging sequence $\{V_k\}$ of estimates of V^* by iterating

Algorithm 1 (Exact policy iteration):

```
1: Choose any $u_0 \in U(X) \bigcap \{u : V(x_0,u) < \infty\}$
2: **repeat**
3:     $V_k = T(u_k)$
4:     $u_{k+1} = H(V_k)$
5: **until** convergence
```

There are at least two problems with this iteration. First, to start the iteration we must provide a controller which stabilizes the system on X, this can be very hard for nonlinear systems. Also, it is in general impossible to find an exact solution in step 3, therefore any computational method should keep track of the successive errors. In the next section we propose such an approximate iteration.

III. Approximate policy iteration

In this section we introduce an alternative to exact policy iteration. The basic idea is to replace equation (7) with two inequalities. It turns out that this simplifies later computations. Moreover, given a function that satisfies these inequalities we obtain a measure of how close this function is to the optimal value function.

A. Basic inequalities

Theorem 1: Given $u \in U(X)$. Let $W \subset X$ be an invariant set of points x such that $\lim_{t\to\infty} \phi(t,x,u) = 0$. Suppose that $\underline{\alpha}, \overline{\alpha} \in \mathbb{R}$ and $V \in C^1(X,\mathbb{R})$ satisfies

$$\frac{\partial V(x)}{\partial x}^T (f(x) + g(x)u(x)) + \underline{\alpha}(l(x) + |u(x)|_R^2) \leq 0 \tag{9}$$

$$0 \leq \frac{\partial V(x)}{\partial x}^T (f(x) + g(x)\hat{u}) + \overline{\alpha}(l(x) + |\hat{u}|_R^2), \quad \forall \hat{u} \in U \tag{10}$$

for all $x \in W$, then

$$\underline{\alpha} V^* \leq V(x) \leq \overline{\alpha} V^*(x), \quad \forall x \in W$$

Proof: Integrating the inequality in (10) along $\phi(t,x,u)$ and using that $\lim_{t\to\infty} V(\phi(t,x,u)) = 0$, we have

$$V(x) \leq \overline{\alpha} \int_0^\infty l(\phi(t,x,u)) + |u(\phi(t,x,u))|_R^2 dt$$

Since this holds for all u we must have

$$V(x) \leq \overline{\alpha} \inf_u \int_0^\infty l(\phi(t,x,u)) + |u(\phi(t,x,u))|_R^2 dt = \overline{\alpha} V^*(x)$$

The other inequality follows in the same way. ■

Theorem 2: Let V be as in Theorem 1. If V is bounded on W then

$$u_+(x) = \arg\min_u \{ \frac{\partial V(x)}{\partial x}^T (f(x) + g(x)u) + \underline{\alpha}(l(x) + |u|_R^2) \}$$

defines a stabilizing cost improving controller. Moreover,

$$\underline{\alpha} V^*(x) \leq \underline{\alpha} V(x, u_+) \leq V(x) \leq \overline{\alpha} V^*(x) \quad \forall x \in W$$

Proof: By (9)

$$\frac{\partial V(x)}{\partial x}^T (f(x) + g(x)u_+(x)) + \underline{\alpha}(l(x) + |u_+(x)|_R^2) \leq 0$$

Integrating along $\phi(t,x,u_+)$ we have

$$\lim_{t\to\infty} V(\phi(t,x,u_+)) + \underline{\alpha} \int_0^\infty l(\phi(t,x,u_+)) + |u(\phi(t,x,u_+))|_R^2 dt \leq V(x)$$

Since V is bounded $\lim_{t\to\infty} \phi(t,x,u_+) = 0$, thus

$$\underline{\alpha} V^*(x) \leq \underline{\alpha} V(x, u_+) \leq V(x)$$

■

The next observation states that if we have two functions V_1 and V_2 satisfying Theorem 1 on W_1, with one better than the other. Then the best approximation is valid on a strictly larger set W_2, moreover the performance achieved on $W_2 \setminus W_1$ is at least as good as that obtained with the policy with the lowest performance.

Theorem 3: Assume that W_1 has C^1 boundary, $u_1, u_2 \in U(X)$ and that V_1, V_2 satisfy (9) and (10) with $\underline{\alpha}_1, \underline{\alpha}_2$, and $\overline{\alpha}_1, \overline{\alpha}_2$. If

$$\underline{\alpha}_1 < \underline{\alpha}_2 \text{ and } \overline{\alpha}_1 > \overline{\alpha}_2 \tag{11}$$

then there exists a $W_2 \subset X$ such that $W_1 \subset W_2$ with strict inclusion, and

$$\underline{\alpha}_1 V^*(x) \leq V_2(x) \leq \overline{\alpha}_1 V^*(x)$$

$$\frac{\partial V_2(x)}{\partial x}^T (f(x) + g(x)u_2(x)) + \underline{\alpha}_1(l(x) + |u_2(x)|_R^2) \leq 0$$

$$\frac{\partial V_2(x)}{\partial x}^T (f(x) + g(x)u_2(x)) + \overline{\alpha}_1(l(x) + |u_2(x)|_R^2) \geq 0$$

for $x \in W_2$.

Proof: Let $y \in \partial W_1$ be a point in the boundary of W_1 and $\eta(y)$ be the outward unit normal at y. We have

$$V_2(y) \geq \underline{\alpha}_2 V^*(y) > \underline{\alpha}_1 V^*(y)$$

By assumption V_2 is continuous on any neighborhood to the boundary, hence there exist a $h_L > 0$ such that for

$$p_L = y + \eta(y) h_L$$

it holds

$$V_2(p_L) > \underline{\alpha}_1 V^*(p_L), \quad \forall y \in \partial W_1$$

Similarly, we can find $h_U, t_L, t_U > 0$ and

$$p_U = y + \eta(y) h_U$$
$$t_L = y + \eta(y) r_L$$
$$t_U = y + \eta(y) r_U$$

such that $\forall y \in \partial W_1$

$$V_2(p_U) < \overline{\alpha}_1 V^*(p_U)$$
$$\frac{\partial V(r_L)}{\partial x}^T (f(r_L) + g(r_L)u(r_L)) + \underline{\alpha}_1(l(r_L) + |u(r_L)|_R^2) < 0$$
$$\frac{\partial V(r_U)}{\partial x}^T (f(r_U) + g(r_U)u(r_U)) + \overline{\alpha}_1(l(r_U) + |u(r_U)|_R^2) > 0$$

We denote by $B(x,r)$ the ball around x with radius r. Let $h = \min\{h_L, h_U, t_L\}$ and take

$$W_2 = W_1 \bigcup \overline{\cup_{y\in\partial W_1} B(y, \eta(y)h)}$$

With this choice the result follows. ■

B. An algorithm

The exact value function belongs to an infinite dimensional space. As a first approximation we constrain the search for an approximation to V^* to a subspace $H_r \subset C^1(X, \mathbb{R})$ of dimension r. Denote by $P(u,r,W)$ the following optimization program:

$$P(u,r,W) \mapsto \min \varepsilon$$

such that $\forall x \in W$

$$\frac{\partial V(x)}{\partial x}^T (f(x) + g(x)u(x)) + \underline{\alpha}(l(x) + |u(x)|_R^2) \leq 0$$
$$\frac{\partial V(x)}{\partial x}^T (f(x) + g(x)u) + \overline{\alpha}(l(x) + |u|_R^2) \geq 0 \quad \forall u \in U$$
$$\overline{\alpha} = 1 + \varepsilon$$
$$\underline{\alpha} = 1 - \varepsilon$$
$$V \geq 0 \text{ and } V(0) = 0$$
$$V \in H_r$$

the solution is given by a function V and the minimal ε for which such a function exists. The computationally most expensive part of the algorithm below is to obtain a solution to $P(u,r,W)$. In general this problem is not easy to solve. However if the system dynamics is modeled with polynomials and H_r is spanned by a polynomial basis the solution can be obtained in a computationally tractable way. Now consider the following pseudo code, *Algorithm 2 (Approximate policy iteration):*

```
1:  Let $u_0$ be any locally stabilizing feedback for system
    (1). Let $W_0$ be an invariant subset of the resulting closed
    loop, with $0 \in W_0$.
2:  repeat
3:    repeat
4:      Solve $P(u_k, r_k, W_j)$.
5:      if $\varepsilon_k$ does not decrease then
6:        increase $r_k$
7:      end if
8:      $u_{k+1} = H(V_k)$
9:    until $\varepsilon_k$ is small
10:   Find some invariant set $W_{j+1} \supset W_j$
11: until $\varepsilon_k$ is small enough and $X \subset W_j$
```

The first step consists of initialization. For example if the system is linearizable and controllable at the origin, we can easily find the corresponding LQ-controller. In step $(3)-(9)$ we hope to obtain a sufficiently small value of ε_k, without having to increase r_k too much. As it increases the computational burden soon becomes excessive, as we explain below. In the last step we try to expand the valid region of approximation, such a set is likely to exist according to Theorem 3.

IV. Solving problem $P(u,r,W)$

In this section we focus on problem $P(u,r,W)$. The computational method relies on representation of positive multivariate polynomials. We start with a brief explanation of the methods, for further information see [11].

A. Sum of squares of polynomials

Consider a multivariate polynomial $f(x) \in \mathbb{R}[x]$ of degree at most $2d$, with $x \in \mathbb{R}^n$. The first, trivial, observation is that if f is a sum of squares $f(x) = \sum_{k=1}^m f_k^2(x)$ for some f_k's of degree at most d, then $f \geq 0$ for all $x \in \mathbb{R}^n$. The following proposition characterizes all such polynomials

Proposition 3: Let $Z(x)$ be a vector of all monomials of degree at most d then f is a sum of squares if and only if

$$f(x) = Z(x)^T Q Z(x) \tag{12}$$

for some positive semidefinite (psd) matrix Q.

Proof: See, [13] ■

The main point now is that it is easy, from a computational point of view, to check if a given polynomial is a sum of squares, which was noted in [12]. Given a polynomial f, checking if f is sum of squares can be done using semidefinite programming as follows: Identify coefficients in (12), this gives an affine constraint on Q, taking the intersection with the convex cone of psd matrices results in a convex constraint. Note also that if the coefficients in f are not predetermined but can be chosen from some affine set, the problem is of exactly the same structure, and this will be important below. For more information see, [15], [11]. The other implication is false; if $f \geq 0$ then f is not necessarily a sum of squares. Thus the above procedure gives sufficient conditions for positivity on $\mathbb{R}^n$.

In this paper we focus on positivity on compact sets. There are several such conditions. Consider a set

$$X = \{h_k(x) \geq 0, k = 1..m\} \tag{13}$$

with all h_k's polynomials. Let us denote by $\Sigma[x]$ the set of sum of squares in $\mathbb{R}^n$ and

$$G_X = \{p : p = s_0(x) + \sum_{k=1}^m s_k(x) h_k(x), \quad s_k \in \Sigma[x]\} \tag{14}$$

The following result will be useful

Theorem 4 (Putinar[14]): Let X be as in eqn.(13). Suppose that there is a real number $r > 0$ such that $r^2 - \sum_{k=1}^n x_k^2 \in G_X$, then for every $f > 0$ on X, $f \in G_X$.

Statement (4) gives necessary and sufficient conditions for f being positive on X, as opposed to above.

B. Solution of $P(u,r,W)$

To be able to apply propositions (4) and (3) to solve $P(u,r,W)$ we assume that the system is modeled by polynomials, $f, g \in \mathbb{R}[x]$ with $x \in \mathbb{R}^n$. Let $H_r^d \subset \mathbb{R}[x]$ be a subspace spanned by a subset of r basis functions of degree at most d. For notational simplicity, let us assume that W can be described with only one polynomial, for example a closed ball centered at the origin with radius a, i.e. $W = \{x : a^2 - |x|^2 \geq 0\}$. We now restate $P(u,r,W)$ as

$$P(u,r,W) \mapsto \min \varepsilon$$

such that

$$\begin{aligned}
&-\left(\frac{\partial V(x)}{\partial x}^T (f(x) + g(x)u(x)) + \underline{\alpha}(l(x) + |u(x)|_R^2)\right) \\
&-s_L(a - |x|^2) \in \Sigma[x] \\
&\frac{\partial V(x)}{\partial x}^T (f(x) + g(x)u) + \overline{\alpha}(l(x) + |u|_R^2) \\
&-s_U(a - |x|^2) \in \Sigma[x,u] \\
&V \in H_r^d \\
&\underline{\alpha} = 1 - \varepsilon \\
&\overline{\alpha} = 1 + \varepsilon \\
&s_L \in \Sigma[x] \\
&s_U \in \Sigma[x,u]
\end{aligned}$$

By the discussion above this is a semidefinite program, it can be solved efficiently as long as the dimension of H_r^d is not too high. The dimension needed in a particular application depends on the required accuracy.

V. Examples

In this section we give two examples of the proposed method. The first example is simple in the sense that we can obtain an analytic solution to the HJB-equation for comparison. The second example is adopted from [2], it originates from [6]. In [2] the authors compare some of the existing methods for computation of suboptimal controllers for nonlinear systems.

The computations in both examples were performed on a standard PC running Matlab.

A. Example 1

Consider the one dimensional linear system

$$\dot{x} = -x + u$$

with non-quadratic instantaneous cost

$$q(x,u) = x^2 + x^4 + u^2$$

The exact solution is readily seen to be

$$V^*(x) = -x^2 + 2\frac{(2+x^2)^{3/2} - 2\sqrt{2}}{3}$$

and

$$u^*(x) = x - x\sqrt{2+x^2}$$

To initialize, take $u_0 = -x$. We would like to find an approximation on $X = [-1,1]$. Since u_0 stabilizes the system on X we don't need to successively expand the region of attraction. To compare, consider the Taylor expansion of the exact solution at the origin

$$V^*(x) = 0.41421x^2 + 0.176776x^4 - 0.0147314x^6 + O(x^7)$$

and

$$u^*(x) = -0.41421x - 0.353552x^3 + 0.0441941x^5 + O(x^6)$$

We successively compute approximations to the value function with degree 2,4 and 6.

$$V_2 = 0.5324x^2$$
$$V_4 = 0.4172x^2 + 0.16006x^4$$
$$V_6 = 0.41437x^2 + 0.175x^4 - 0.010862x^6$$

with associated controls

$$u_2 = -0.5337x$$
$$u_4 = -0.4178x - 0.32102x^3$$
$$u_6 = -0.41437x - 0.35x^3 + 0.032574x^5$$

The reason for the differences between the computed approximations and the Taylor expansion is that our method computes a uniform approximation whereas the Taylor expansion is a local approximation. As expected this difference is most notable for the second degree approximation. For the sixth degree approximation we have the following estimate

$$\max_{x\in X} |V^*(x) - V_6(x)| \approx 10^{-2} \max_{x\in X} |V^*(x) - V_T(x)|$$

where V_T is the sixth degree Taylor polynomial. Although this is a simple example, it gives some indication of the usefulness of our method as compared to power series expansion methods. The computations are summarized in Table I . The leftmost column, "Deg", shows the degree of the approximation, "#SDP" shows the total number of semidefinite programs solved to obtain the approximation, "CPU(s)" is the total execution time in seconds and finally "$\varepsilon(\%)$" shows the uniform relativ distance, in percentage, to the exact value function. Notice that, for example, the policy obtained from the second degree approximation is used to initialize the algorithm to obtain the forth degree approximation, so that only two new semidefinit programs were solved in the second step, the same holds for the computation time.

TABLE I
RESULTS FOR EXAMPLE 1

Deg	#SDP	CPU(s)	$\varepsilon(\%)$
2	2	3.2	28.8
4	4	6.2	0.720
6	6	10.6	0.0364

TABLE II
RESULTS FOR EXAMPLE 2

Deg	#SDP	CPU(s)	$V(x_0)$	$\varepsilon(\%)$
2	3	170	0.0436	91.6
3	5	270	0.0412	67.6
4	7	370	0.0655	21.4

B. Example 2

In this example we consider a flight control problem. The system is modeled with three states x_1 ="angle of attack", x_2 ="flight path angle" and x_3 ="rate of change of flight path angle". The control variable u is the tail deflection angle. The states and control variable should be interpreted as deviations from some setpoint. The model is as follows

$$\begin{aligned} f_1(x) =& -0.877x_1 + x_3 + 0.47x_1^2 - 0.088x_1x_3 - 0.019x_2^2 \\ & + 3.846x_1^3 - x_1^2x_3 \\ f_2(x) =& x_3 \\ f_3(x) =& -4.208x_1 - 0.396x_3 - 0.47x_1^2 - 3.564x_1^3 \end{aligned}$$

and

$$g^T = [-0.215 \quad 0 \quad -20.967]$$

In this example the instantaneous cost is $q(x,u) = 0.25x^Tx + u^2$. The region were the approximation should be valid is $X = B[0, 25(\pi/180)]$. To evaluate suboptimal policies we will compare their ability to bring the system to rest after an initial perturbation in the angle of attack, $x_0^T = [25(\pi/180) \quad 0 \quad 0]$.

The algorithm was initialized with $W_0 = B[0,0.2]$ and u_0 was the LQ-controller for the linearized system. The second iteration produced a linear controller that stabilizes the system on $W = B[0, 25(\pi/180) + 0.1]$, which was then used as the valid region of approximation for the rest of the iterations. The results are summarized in Table II, figure (1) and (2). The remark on the number of iterations and execution time in example 1 also applies in this example. These results can be compared to the results obtained in [2] for the same problem. The best controller for this problem were obtained with a discretization-interpolation method. The performance of that controller is about the same as that we obtained with the third degree approximation. The reported computation time required for the discretization-interpolation method is about 6000 seconds, on a similar Matlab implementation, compared to the method in this paper which requires 370 seconds. The difference depends, of course, not only on the methodology but also on the software. Moreover, our controller is a third

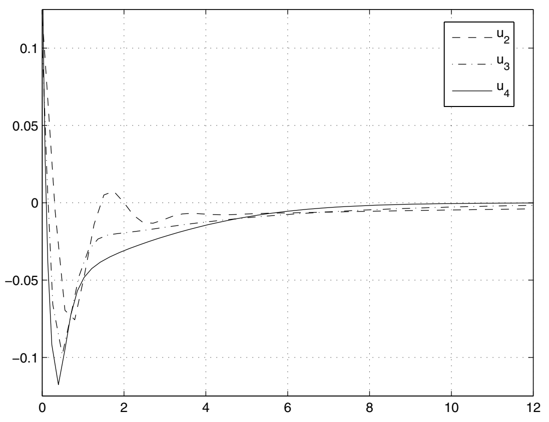

Fig. 1. Control signal for different degrees of approximation in example 2

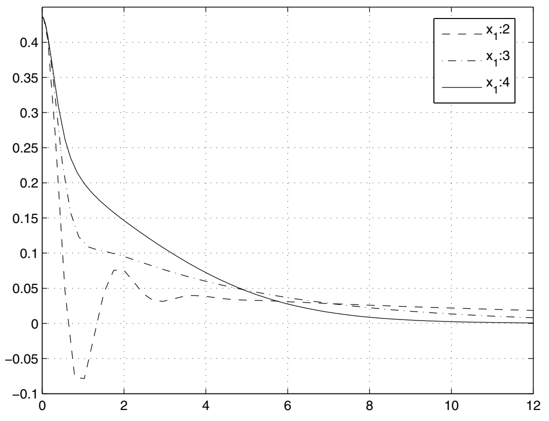

Fig. 2. State x_1 for different degrees of approximation in example 2.

degree polynomial, to be compared with the complicated representation for the discretization-interpolation method.

VI. CONCLUSIONS

We proposed a new algorithm for feedback nonlinear synthesis. The method gives bounds on suboptimality, as measured form the optimal value function. We have showed that there is a tradeoff between the size of the region where the approximation is valid and how close the approximation is to the exact value function.

For systems that are modeled with polynomials the required computations can be done in a tractable way via convex optimization.

REFERENCES

[1] R. Beard, G. N. Saridis and J. T. Wen, Approximate Solutions to the Time-Invariant Hamilton-Jacobi-Bellman Equation, *J. of Optimization Theory And Applications* vol. 96, No. 3, pp. 589-626, March 1998.

[2] S. C. Beeler, H. T. Tran and H. T. Banks, Feedback Control Methodologies, *J. of Optimization Theory and Application*, vol. 107, No. 1, pp. 1-33, 2000

[3] R. Bellman, Dynamic Programming, *Princeton University Press*, 1957

[4] W. H. Fleming, H. M. Soner, Controlled Markov Processes and Viscosity Solutions, *Springer, Applications of Mathematics*, 25, 1993

[5] W.L. Garrard, Additional results on suboptimal feedback control of nonlinear systems, *Int. J. of Control*, 10(6), pp. 657-663

[6] W.L. Garrard and J.M. Jordan, Design of nonlinear automatic flight control systems, *Automatica*, vol. 13, pp. 497-505, 1977

[7] Y. Huang and W. M. Lu, Nonlinear optimal control: Alternatives to Hamilton-Jacobi equation, *In Proc. IEEE Conf. on Decision and Control*, 1996

[8] R. J. Leake and Ruey-wen Liu, Construction of suboptimal control sequences, *SIAM J. Control and Optimization* vol. 5, No. 1, pp. 54-63, 1967.

[9] J. Markman and I.N. Katz, An Iterative Algorithm for Solving Hamilton-Jacobi Equations, *SIAM J. Sci. Comput.* 22, pp. 312-329, 2000

[10] Nishikawa, Y., N. Sannomiya and H. Itakura, A method for suboptimal design of nonlinear feedback systems.*Automatica*, 7, pp.703-712, 1971

[11] P.A. Parrilo, Semidefinite programming relaxations for semialgebraic problems.*Mathematical Programming Ser. B*, Vol. 96, No.2, pp. 293-320, 2003

[12] P.A. Parrilo, Structured Semidefinite Programs and Semialgebraic Geometry Methods in Robustness and Optimization, *PhD thesis, California Institute of Technology, Pasadena, CA*, May 2000.

[13] V. Powers and T. Wormann, An algorithm for sums of squares of real polynomials, *J. Pure and Applied Algebra* 127 (1998) 99-104

[14] M. Putinar, Positive polynomials on compact semi-algebraic sets, *Indiana Univ. Math. J.* 42, No. 3,pp. 969-984, 1993

[15] S. Prajna, A. Papachristodoulou and P. Parrilo, Introducing SOS-TOOLS: A general purpose sum of squares programming solver, *Proc. IEEE Conf. on Decision and Control* 2002

[16] S. Prajna, A. Papachristodoulou and F. Wu, Nonlinear Control Synthesis by Sum of Squares Optimization: A Lyapunov-based Approach, *Proceedings of the Asian Control Conference (ASCC), Melbourne, Australia.* 2004

[17] L. Vandenberghe and S. Boyd, Semidefinite programming, *SIAM Review, vol. 38, No. 1, pp 49-95* 1996