AN IMPROVED SPSA ALGORITHM FOR STOCHASTIC OPTIMIZATION WITH BOUND CONSTRAINTS [1]

Dobrivoje Popović Andrew Teel [**]
Mrdjan Janković [***]

[*] *United Technologies Research Center, 411 Silver Lane MS129-15, East Hartford, CT 06118 e-mail:{popovidi}@utrc.utc.com.*
[**] *Department of Electrical and Computer Engineering, University of California Santa Barbara, Santa Barbara, CA 93106-9560, e-mail:{teel}@ece.ucsb.edu.*
[***] *Ford Motor Company, Research & Advanced Engineering, P.O. Box 2053, MD 2036 SRL, Dearborn, MI 48121, e-mail: mjankov1@ford.com.*

Abstract: We show that the Simultaneous Perturbation Stochastic Approximation (SPSA) algorithm with projection may exhibit slow convergence in constrained stochastic optimization problems when the optimum is situated on the constraints. The cause of the slow convergence is a geometric interaction between the projection operator and the SPSA gradient estimate. The effect of this interaction can be described as "bouncing of iterates against the constraints." We describe this on two low dimensional noise-free examples, and present a new algorithm that does not exhibit the bouncing effect and the consequent slow convergence.

Keywords: Stochastic approximation, algorithms, real-time, parameter optimization, constrained parameters, engine efficiency.

1. INTRODUCTION

SPSA algorithm (**?**) is one of the most popular algorithms that uses finite difference gradient estimation for stochastic optimization problems. The main reason for this popularity is its efficient gradient estimation: a typical SPSA implementation uses only two measurements of the cost function per iteration. In contrast, classic Kiefer-Wolfowitz-Blum Stochastic Approximation (SA) requires $2p$ measurements, p being the number of parameters (**?**). This measurement frugality of SPSA indeed translates into a p-fold saving in the total number of measurements needed for optimization (**?**).

The work described here is motivated by an application of SPSA to on-line optimization of automotive internal combustion engine (**?**). Mathematically, this is as an optimization problem with bound constraints: $\min_{x \in G} f(x)$, where $G = \{x = [x_1, x_2, ..., x_p]^\top \in \mathbb{R}^p$, with $a_i \leq x_i \leq b_i$, $i = 1, 2, ..., p$ such that $-\infty < a_i < b_i < \infty$, $i = 1, 2, \ldots, p$. Here, the constraints are hard, i.e. noisy evaluations of the cost function $f : \mathbb{R}^p \to \mathbb{R}$ are available only over the set G. The problem is a perfect setting for SA algorithms with projection

[1] An extended version of this paper is submitted to Automatica.

- in the case of bound constraints, projection of the iterates is a simple truncation.

An SA algorithm with projection can be described as follows: Let the iterate x_k be the kth estimate of the optimal parameter values, $g(x_k)$ be the gradient of $f(\cdot)$ at x_k, and $\widehat{g}_k(x_k)$ be an estimate of $g(x_k)$. Then, $x_{k+1} = \pi_G\{x_k - a_k\widehat{g}_k(x_k)\}$, where the step size $a_k \downarrow 0$ as $k \to \infty$, and $\pi_G : \mathbb{R}^p \to \mathbb{R}^p$ is the projection onto the set G, see for example (**?**) for general SA algorithms and (**?**) for SPSA.

In our application (see §4), SPSA with projection turned out to be significantly faster in reaching the minimum than the classic SA with projection. However, this was only after the SPSA was modified. We initially encountered problems that the theory and previous experience did not suggest. The problems are caused by an interesting geometric interaction between the projection operator and the SPSA gradient estimate. This interaction happens when one or more components of the current iterate reach the boundary of the constrained set while the (negative) gradient field is directed against the boundary. The effect of this interaction can be described as "iterate bouncing against the constraints:" the iterates first reach the boundary of the constrained set; then, in the next iteration they "bounce" back, and the value of the cost function increases (see Figure **??** for a quick visual). The next iteration then forces the iterates back towards the boundary, and they bounce again. We will describe this in detail later.

When the iterates are far away from the optimum, the influence of the bouncing on optimization speed is unclear: we encountered some situations where the optimization speed was improved, and some where it was slowed down. However, consider the situation when the iterates reached a neighborhood of the optimum, and the minimum belongs to the boundary of the constrained set. In this case, it can be expected that the iterates produced by an algorithm with projection will be frequently located on the boundary during the rest of the optimization process. As a consequence, the bouncing will then be happening with an increased frequency as well; this time, however, iterates will be bouncing around the optimum. The consequence will be a slower convergence.

In our engine application, as we implied, we had a case of the minimum located on the boundary of the constrained set. The iterates usually approached the minimum quickly, but the bouncing then slowed down the convergence. In this situation, a sufficient reduction of the step-size would bring the bouncing to acceptable levels. But the step-size would need to be reduced much more (and the optimization slowed down more) than the noise in the problem actually required. In applications that use constant step-size, such as tracking of a time-varying optimum, reduction of the step-size sometimes may not even be an option.

The goal of this paper is to explain the bouncing and introduce a simple modification to the SPSA algorithm that does not suffer from it. This modification has the following properties: 1) Any user-provided SPSA algorithm can easily be upgraded. 2) When all of the elements of x_k are inside the constrained set, it performs the same iteration as would the original SPSA algorithm. 3) It requires the same number of measurements per iteration as the original algorithm. 4) Its theoretical convergence properties can easily be established.

Other SA algorithms; more general types of constraints: In general, an algorithm that estimates the gradient from just one finite difference will exhibit the bouncing when constraints are present and projection is used. Apart from SPSA, the class of these algorithms includes, for example, Random Direction Stochastic Approximation (RDSA) algorithm by Kushner and Clark, (**?**, p.58) for a unified treatment of all these algorithms see (**?**). In the case of bound constraints, these algorithms can be modified in a way very similar to the one described here.

2. PROJECTION SPSA ALGORITHM AND BOUND CONSTRAINTS

At a point x_k, SPSA estimates the gradient vector $g(x_k)$ of the function $f(\cdot)$ as follows (**?**):

(1) It randomly generates the perturbation vector $\Lambda_k = [\Delta_{k1}\Delta_{k2}\dots\ \Delta_{kp}]^\top \in \mathbb{R}^p$, where $\Delta_{ki}, i = 1, 2, ..., p$ are mutually independent, symmetric Bernoulli distributed (this is just one possibility, though the most popular).

(2) Then, it collects two possibly noisy measurements of the cost function: $y_k^{(+)}=f(x_k + c_k\Lambda_k)+\eta_k^{(+)}$ $y_k^{(-)}=f(x_k - c_k\Lambda_k)+\eta_k^{(-)}$ where $\eta_k^{(+)}$ and $\eta_k^{(-)}$ denote the noise, and c_k is a small positive number. It is said that the parameters are simultaneously perturbed in $x_k \pm c_k\Lambda_k$ since each component of Λ_k is nonzero.

(3) It produces each component of the gradient estimate as

$$\hat{g}_{km}(x_k) = \frac{y_k^{(+)} - y_k^{(-)}}{2c_k\Delta_{km}},\ m = 1, 2, ..., p \quad (1)$$

Possibly the two most important features of SPSA are that it produces an estimate of the gradient from only two measurements, and that each measurement corresponds to a simultaneous perturbation of every component of x_k. These features are the main causes of the efficiency of SPSA. As we will see in the following elementary example,

however, they are also responsible for the bouncing effect.

Example 1. **A simple noise-free optimization problem with bound constraints.** Consider

$$\min_{x \in G} f(x) = x_1^2 + x_2^2$$

with $x = [x_1, x_2]^\top$, $G = \{x \in \mathbb{R}^2 | \ x_2 \geq 1\}$. The minimum is achieved for $x^* = [0\ 1]$, with $f(x^*) = 1$.

Take the following algorithm:

$$\begin{aligned} \tilde{x}_{k+1} &= \\ &= x_k - a \begin{bmatrix} \frac{1}{\Lambda_{k1}} \\ \frac{1}{\Lambda_{k2}} \end{bmatrix} \frac{f(x_k + c\Lambda_k) - f(x_k - c\Lambda_k)}{2c} \\ x_{k+1} &= \pi_G\{\tilde{x}_{k+1}\} \end{aligned} \tag{2}$$

where $a = 4,\ c = 0.1$, a and c are constant for simplicity. Perturbation vector Λ_k can be either $[1\ 1]^\top$ or $[1\ -1]^\top$, with equal probability of $\frac{1}{2}$.

We next show the first three iterations of this algorithm starting from $x_0 = x^*$; we assume that the first three perturbation vectors are generated as $\Lambda_1 = \Lambda_2 = \Lambda_3 = [1\ \ -1]^\top$, an event that can happen with probability $\frac{1}{8}$. These three iterations are shown in Figure 1. We get:

$\tilde{x}_1 = [0.40\ 0.60]^\top$, $f(\tilde{x}_1) = 0.52$, $x_1 = [0.40\ 1]^\top$, $f(x_1) = 1.16$, $\tilde{x}_2 = [0.64\ 0.76]^\top$, $f(\tilde{x}_2) = 0.99$, $x_2 = [0.64\ 1]^\top$, $f(x_2) = 1.41$, $\tilde{x}_3 = [0.78\ 0.86]^\top$, $f(\tilde{x}_3) = 1.35$, $x_3 = [0.78\ 1]^\top$, $f(x_3) = 1.61$. (For now, assume that the experiments to measure $f(\cdot)$ can be performed even if $x_k + c\Lambda_k$ is outside of G). So, for $k = 0, 1, 2$ we have $f(\tilde{x}_{k+1}) < f(x_k)$, but

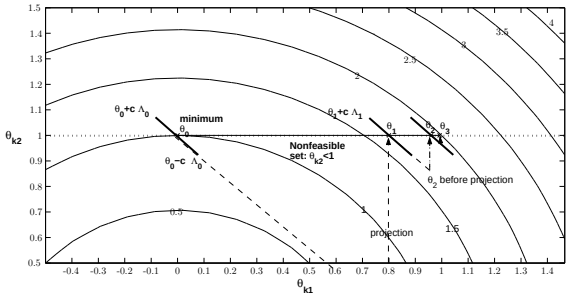

Fig. 1. Minimizing $x_{k1}^2 + x_{k2}^2,\ x_{k2} \geq 1$ with the algorithm (2); the three iterations are shown, starting from the minimum $x_0 = x^* = [0\ 1]^\top$. Perturbation vectors are $\Lambda_1 = \Lambda_2 = \Lambda_3 = [1\ \ -1]^\top$.

$f(x_{k+1}) > f(x_k)$. The iterates wander away from the optimum. Note that effects would be similar if $\Lambda_1 = \Lambda_2 = \Lambda_3 = [1\ 1]^\top$, which can also happen with probability $\frac{1}{8}$ ■

What is the cause of this? First, we have $f(x_k + c\Lambda_k) - f(x_k - c\Lambda_k) \approx 2c\ g(x_k)^\top \Lambda_k$, and around the optimum, $2c\ g(x_k)^\top \Lambda_k$ is primarily produced by the components of the gradient corresponding to the parameters located on the boundary - in the above example that is $\frac{\partial f}{\partial x_2}$ ($\frac{\partial f}{\partial x_1}$ is close to zero around the minimum $[0\ 1]^\top$). Now, SPSA estimates each component of the gradient from the same finite difference; from (1) it is obvious that the less dominant components of the gradient can be estimated with the wrong sign (especially so around a minimum on the boundary). Consequently, the corresponding components of x_k can be updated in the wrong direction. When there are no constraints, this doesn't matter, since x_{k+1} is, in general, still better than x_k. However, when the constraints are present as in our example, the updates of the elements of x_k that would produce the improvement in $f(\cdot)$ may end up truncated by the projection. At the same time the (wrongly calculated) updates to the less dominant elements may still be kept.

This may produce variations of the iterates around the optimum, as it can be seen in Figure **??**. This is a prolonged version simulation of the Example 1, but now Λ_k varies randomly.

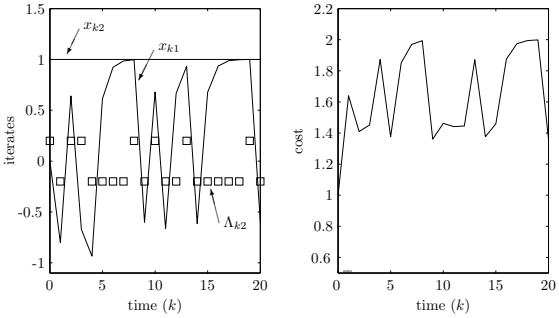

Fig. 2. Variation of the iterates and the cost function during minimization of $x_{k1}^2 + x_{k2}^2,\ x_{k2} \geq 1$ with SPSA. The perturbation vectors Λ_k are randomly chosen between $[1\ 1]^\top$ and $[1\ \ -1]^\top$; only Λ_{k2} is shown (not to scale: $\Lambda_{k2} = \pm 1$).

When each component of the gradient is estimated from a separate finite difference, projection will not produce bouncing in the case with bound constraints. Therefore, one solution to eliminate bouncing would be to switch from SPSA to full gradient estimation when the iterates reach the boundary, and use p finite-differences to estimate p gradient components. This can be an expensive solution (which we demonstrate in the full version of this paper), For, what if we have many parameters, but just one or two elements of the current iterate x_k are on the boundary of the constraint set? The idea is to appropriately change the pool of perturbation vectors. In the simplest case represented by Example 1, the algorithm select Λ_k between $[0\ 1]^\top$ and $[1\ 0]^\top$ when x_{k2} is on the boundary. The simulation outcome shown

in Figure **??**. We proceed to describe the new

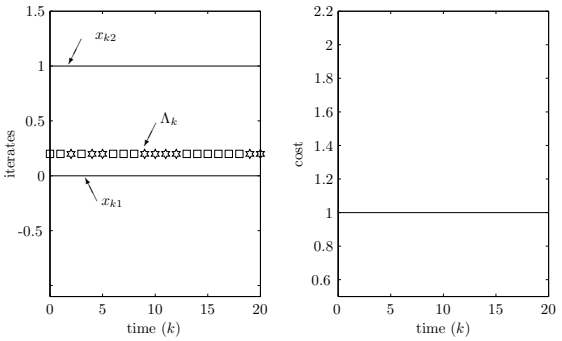

Fig. 3. Minimization with the modified SPSA (Section 3, Algorithm 1), c.f. Figure **??**.

algorithm.

3. MODIFICATION TO PROJECTION SPSA FOR BOUND CONSTRAINTS

The new algorithm below, like SPSA, uses two measurements and one finite difference to estimate the gradient. However, at each iteration, it first separates those $x_{ki},\ i = 1, ..., p$, that are on the boundary of the feasible set G from the ones that belong to the interior of G. Then, it randomly decides whether it will perturb just one of x_{ki} located on the boundary, or simultaneously perturb all of those that are in the interior of G. This additional randomization is such that it produces an unbiased estimate of the gradient. This is implemented via the following modification of SPSA: At each iteration, the algorithm first generates an additional random variable, and then computes indicators that signal which x_{ki} are on the boundary, and which are not. Based on the value of the random variable and the values of the indicators, two simple formulas alter the original SPSA perturbation vector and the SPSA gradient estimate. One iteration of the new algorithm looks as follows (we assume that an SPSA algorithm is already provided by the user):

Algorithm 1. Let k be the current iteration and x_k the current iterate.

(1) Generate positive numbers a_k and c_k, as in the original SPSA algorithm.

(2) Randomly generate perturbations Δ_{k1}, Δ_{k2},..., Δ_{kp} as in the original SPSA algorithm.

(3) Generate a discrete random variable H_k that assumes a value from the set $\{1, 2, ..., p\}$ with uniform probability equal to $\frac{1}{p}$. Compute the switching variables $\kappa_{ki},\ i = 1, 2, ..., p$:

$$\kappa_{ki} = \begin{cases} 1, & \text{if } H_k = i \\ 0, & \text{otherwise .} \end{cases}$$

(4) Let $\delta_i(x_k), \imath = 1, 2, ...p$ be an indicator that signals when x_{ki} is on the boundary of G:

$$\delta_i(x_k) = \begin{cases} 0,\ a_i < x_k < b_i \\ 1,\ x_k = a_i \text{ or } x_k = b_i \end{cases} .$$

We will use shorthand notation $\delta_{ki} := \delta_i(x_k)$.

(5) Now, produce the perturbation vector Λ_k :

$$\Lambda_k = \left(\sum_{i=1}^{p} \kappa_{ki}(1 - \delta_{ki}) \right) \begin{bmatrix} (1-\delta_{k1})\Delta_{k1} \\ (1-\delta_{k2})\Delta_{k2} \\ \vdots \\ (1-\delta_{kp})\Delta_{kp} \end{bmatrix} + \begin{bmatrix} \kappa_{k1}\ \delta_{k1}\Delta_{k1} \\ \kappa_{k2}\ \delta_{k2}\Delta_{k2} \\ \vdots \\ \kappa_{kp}\ \delta_{kp}\Delta_{kp} \end{bmatrix} .$$

For example, if only x_{k2} and x_{k4} are on the boundary of G, H, Λ_k can be either $[\Delta_{k1}, 0, \Delta_{k3}, 0, \Delta_{k5}, \Delta_{k6}, \ldots, \Delta_{kp}]^\top$ with probability $\frac{p-2}{p}$, or either $[\,0, \Delta_{k2}, 0, 0, \ldots, 0]^\top$ or $[0, 0, 0, \Delta_{k4}, 0, 0, \ldots, 0]^\top$ with probabilities $\frac{1}{p}$. Recall that p represents the number of optimized parameters.

(6) Collect two measurements, ${y_k}^+$ and ${y_k}^-$:

$$\begin{aligned} {y_k}^+ &= f(x_k + c_k\Lambda_k) + {\eta_k}^+ \\ {y_k}^- &= f(x_k - c_k\Lambda_k) + {\eta_k}^- \ , \end{aligned}$$

where ${\eta_k}^+$ and ${\eta_k}^-$ represent noise.

(7) Estimate each component of the gradient as follows, for $m = 1, 2, ..., p$:

$$\widehat{g}_{km}(x_k) = \Big[\frac{p\ (1-\delta_{km})}{p - \sum_{i=1}^{p} \delta_{ki}} \left(\sum_{i=1}^{p} \kappa_{ki}(1-\delta_{ki}) \right) + p\ \kappa_{km}\ \delta_{km}\Big] \frac{{y_k}^+ - {y_k}^-}{2c_k\Delta_{km}} \ . \tag{3}$$

If all of the estimates are on the boundary the above formula produces division by zero, and the following formula should be used instead:

$$\widehat{g}_{km}(x_k) = p\ \kappa_{km}\ \frac{{y_k}^+ - {y_k}^-}{2c_k\Delta_{km}} \ .$$

Note that when none of the elements of x_k is on the boundary, the term in brackets in formula (3) is equal to one and $\Lambda_k = [\Delta_{k1}\ \Delta_{k2}...\Delta_{kp}]^\top$; hence, the algorithm performs an iteration of the original SPSA algorithm.

(8) Finally, update the iterate:

$$x_{k+1} = \pi_G\{x_k - a_k\hat{g}_k(x_k)\} \ , \tag{4}$$

where $\pi_G\{\cdot\}$ is the projection operator onto the set G.

We now introduce the following assumptions about Algorithm 1.

Assumption 1. Let $\Omega = \{\omega\}$ *be the sample space generating random sequences* $\{\Delta_{ki}\}, i = 1, 2, ..., p$, $\{H_k\}$,$\{\eta_k^+\}$, $\{\eta_k^-\}$, *and* $\{x_k\}$ *(via recursion 4). Assume that for all* $k \geq 0$*:*

(1) $\Delta_k = [\Delta_{k1}, \Delta_{k2} \dots \Delta_{kp}]^\top$ *is such that* Δ_{ki}, $i = 1, ..., p$ *are mutually independent, zero mean random variables, and* $\{\Delta_k\}$ *is a mutually independent sequence, with* Δ_k *independent of* $x_0, x_1, \dots, x_k$*. It also holds that* $|\Delta_{ki}| \leq \alpha_0$*, and* $|\Delta_{ki}|^{-1} \leq \alpha_1$*.*

(2) Random variables H_k, $k = 1, 2, ...$ *are uniformly distributed over the set* $\{1, 2, ..., p\}$ *and mutually independent;* H_k *is independent of* Δ_{ki}, $i = 1, 2, ..., p$*, and also independent of* $x_0, x_1, \dots, x_k$*.*

(3) For the measurement noise, it holds that

$$E\left(\eta^+ - \eta^- \mid x_0, x_1, \dots, x_k, \Delta_k, H_k\right) = 0 \ ,$$

and there exists a positive constant γ *such that* $E\left((\eta^+ - \eta^-)^2\right) \leq \gamma$ *.*

(4) $G := \{x = [x_1, x_2, ..., x_p]^\top \in \mathbb{R}^p \mid a_i \leq x_i \leq b_i, \ i = 1, 2, ..., p\}$ *.*

(5) Function f *is twice continuously differentiable on an open set containing* G^o*, defined as:* $G^o = \{x^o \in \mathbb{R}^p \mid \min_{x \in G} |x^o - x| \leq \alpha_0 \sqrt{p} \}$ *.*

Note that by these assumptions, Algorithm 1 evaluates $f(\cdot)$ only within the domain G^o, which contains G.

We can relax these assumptions to hold almost surely, and for $k \geq K$, $K \geq 0$, as it was done in, for instance, (**?**, Lemma 1). Note also that conditions $|\Delta_{ki}| \leq \alpha_0$, and $|\Delta_{ki}|^{-1} \leq \alpha_1$ are stronger than standard SPSA conditions; these can be relaxed, but are not restrictions because in applications perturbations are controlled by user. The following lemma is stated without proof to save space:

Lemma 2. Let Assumption **??** *hold. Then, there exist positive constants* K_1 *and* K_2 *such that for each component of the gradient estimate (3), i.e. for* $m = 1, 2, ..p$*, and for any* $x_k \in G^o$*, it holds that* $E\left(\widehat{g}_{km}^2(x_k)\right) \leq \frac{K_1}{c_k^2}$ *and* $\left| E\left(\widehat{g}_{km}(x_k) - g_m(x_k) \ \middle| \ x_k\right)\right| \leq K_2 \ c_k$ *.*

Theorem 3. Let Assumptions **??** *and hold for Algorithm 1, the function* $f(\cdot)$*, and the set* G*. Assume that For all* k*,* $a_k > 0$*,* $\lim_{k \to \infty} a_k = 0$*,* $c_k > 0$*, and* $\lim_{k \to \infty} c_k = 0$*. In addition,* $\sum_{k=0}^{\infty} a_k = \infty$*,* $\sum_{k=0}^{\infty} \left(\frac{a_k}{c_k}\right)^2 < \infty$ *.*

Let $\mathcal{C}$ *be the set of all Karush-Kuhn-Tucker points of* $f(\cdot)$ *on* G*. Then, the sequence* x_k *generated by Algorithm 1 converges to the set* $\mathcal{C}$*. with probability one.*

Proof. The proof is similar to the proof of (**?**, Proposition 1) and uses Lemma **??** and (**?**, Theorem 5.3.1); therefore, it is omitted to save space. ■

Modification for hard constraints In our development, we implicitly assumed that we can perform experiments over the set G^o, which strictly contains G. If the constraints are hard, experiments are restricted to G. Appropriate modifications for that case can be adopted from (**?**) in a straightforward way.

4. ON-LINE ENGINE OPTIMIZATION

In this section we present an application of SPSA to an automotive engine optimization problem (**?**). Our engine was a dual-independent variable cam timing (diVCT) engine installed in a Ford dynamometer test cell. In this engine, the control system can on-line and independently change three timing (angle) parameters, which are specified relative the position of the crank-shaft: 1) the timing of the the intake valves opening (*ivo*), 2) the timing of the exhaust valves closing (*evc*), and 3) the spark timing (*spark*). For a given operating point, specified by engine speed and output torque, these parameters need to be optimized to achieve the lowest fuel consumption. The cost function that maps the parameters to the fuel consumption is unknown.

The idea of the on-line optimization (also called extremum seeking in the controls literature) is to have an optimization algorithm, in this case SPSA, in control of the engine parameters. The algorithm then performs the optimization on-line by measuring the cost, and adjusting the parameters towards reducing that cost. The cost here is the steady-state Brake Specific Fuel Consumption (BSFC $= \frac{Fuelflow}{Speed \times Torque}$).

The following details place this optimization problem within the framework of this paper 1) The parameters were constrained by hardware: in this case, *ivo* could be varied within the interval of $[-30...30]$ degrees, and *evc* within $[0...40]$ degrees; [2] 2) for many operating points, the optimal parameter values were located on the boundary of the constrained set. Consequently, it can be expected the bouncing will appear when the standard SPSA is applied, but that it can be eliminated with the modified algorithm. We will demonstrate this with simulations on a on a dynamical model of our engine (for the experimental results, consult (**?**).

[2] These numbers represent the corresponding crankshaft angle; this angle is equal to zero degrees when the piston is at the top dead center.

The simulations are noise-free to make the bouncing effect more clear. Namely, the bouncing looks random due to the inherent SPSA randomness, and in real conditions it may be easily confused for noise.

The engine model represents the engine at 1500 rpm. The model includes the torque-to-throttle feedback mechanism that to any change reacts with readjusting the fuel flow (the air/fuel ratio is constant) to maintain the desired torque. Figure **??** shows a two-dimensional representation of the cost function: BSFC is plotted for different *ivo* and *evc* pairs and their maximum brake torque spark timing (this is the optimal *spark* for the selected *ivo* and *evc*). Note that this BSFC map has two

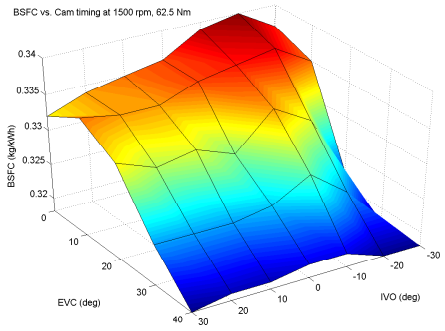

Fig. 4. BSFC of a diVCT engine for optimal spark vs. cam timing (1500 rpm; 62.5 Nm).

local minima which are located on the boundary of the constrained set; in the model they are located at $(ivo, evc) = (-30, 40)$ and $(ivo, evc) = (30, 40)$.

In the first simulation, Figure **??**, the engine is optimized by the standard SPSA algorithm with projection. Note that the effect of "bouncing of the estimates against the constraints" starts around $t \approx 400$, when the estimates reach the neighborhood of the optimum. In the second sim-

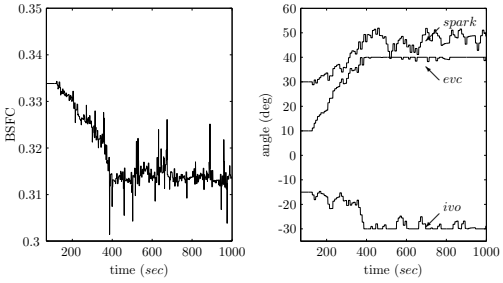

Fig. 5. On-line optimization of the diVCT engine model with standard SPSA with projection ($a_k = const$). Note that the bouncing prevents the iterates from reaching the minimum.

ulation, Figure **??**, the engine is optimized by the modified SPSA algorithm, and the bouncing is eliminated. In both simulations the initial parameter values were $(ivo, evc, spark) = (-15, 10, 30)$, an the sought local minimum is

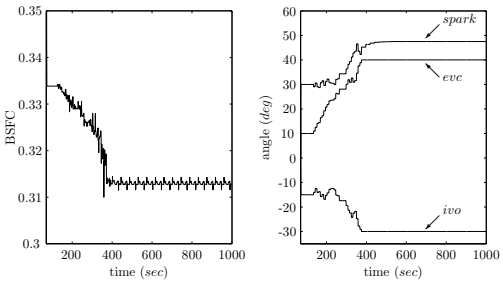

Fig. 6. On-line optimization of the diVCT engine model by the modified SPSA. The bouncing is eliminated, and the iterates converge to the minimum, c.f. Figure **??**.

$\theta^* = (-30, 40, 48)$. Each measurement (two per iteration) was obtained by assigning step commands to the input, and waiting for four seconds for the dynamics to settle down. The step-size was fixed, i.e. $a_k = const$.

5. CONCLUSIONS

In this paper we described an undesirable interaction between the projection operator and the SPSA algorithm. The interaction can appear in constrained stochastic optimization problems, and can slow down the convergence when the optimum lies on the boundary of the constrained set. The effect of this interaction can be described as "bouncing of iterates against the constraints." We showed an engine optimization problem where this bouncing was a serious issue, which had to be resolved. To eliminate bouncing, we designed a modified SPSA algorithm. The modification is created to be simple, and alter the original, user provided, SPSA as little as possible. A modification for more complicated constraints is a subject of our future research.

REFERENCES

Dippon, J. (2003). Accelerated Randomized Stochastic Optimization. *Annals of Statistics*.

Kushner, H. J. and D. S. Clark (1978). *Stochastic Approximation Methods for Constrained and Unconstrained Systems*. Springer-Verlag. New York, NY.

Popović, D., M. Janković, S. Magner and A. Teel (2003). Extremum seeking methods for optimization of variable cam timing engine operation. In: *American Control Conference*. Denver, CO. pp. 3136–3141.

Sadegh, P. (1997). Constrained optimization via stochastic approximation with a simultaneous perturbation gradient approximation. *Automatica* **33**(5), 889–892.

Spall, J. C. (1992). Multivariate stochastic approximation methods. *IEEE Transactions on Automatic Control* **37**(3), 332–341.