SPSA on the Noisy Function Testbed

Steffen Finck
Vorarlberg University of Applied Sciences
Hochschulstrasse 1
Dornbirn, Austria
steffen.finck@fhv.at

ABSTRACT

This paper benchmarks the Simultaneous Perturbation Stochastic Algorithm (SPSA) [4] on the BBOB 2009 noisy testbed. SPSA is a widely used optimization algorithm with its main application in noisy optimization. The paper presents briefly the algorithm and used parameter setting for the testbed.

Categories and Subject Descriptors

G.1.6 [**Numerical Analysis**]: OptimizationGlobal Optimization, Unconstrained Optimization; F.2.1 [**Analysis of Algorithms and Problem Complexity**]: Numerical Algorithms and Problems

General Terms

Algorithms

Keywords

Benchmarking, Black-box optimization, evolutionary computation, stochastic optimization

1. INTRODUCTION

The SPSA algorithm is a very common and widely used optimization algorithm [5] and primarily designed for noisy optimization. In this paper the basic variant with a simple multistart procedure is presented. The main feature of SPSA is the use of just 2 function evaluations to determine the gradient, independent of the search space dimension DIM. As shown in [4, 6] this is advantageous (especially for large DIM) compared with common stochastic approximation algorithm which use $2 \times$ DIM function evaluations to approximate the gradient. The here presented algorithm is coupled with a simple restart procedure to effectively use the given number of maximal function evaluations, similar to Fig. 3 in [2].

2. ALGORITHM PRESENTATION

In Fig. 1 the main algorithm is presented.

The gain `ak` is used for the update of the current search point, while the gain `ck` is used for the test step of the gradient approximation. The determination of their initial values is shown in Fig. 2. To improve the performance of SPSA, `lambda` gradient approximations are averaged within one iteration before the update of the current search point. Thus, the here presented SPSA uses $2 \times$ `lambda` function evaluations per iteration.

To effectively use the allowed number of function evaluations a simple restart procedure was implemented. It is shown in Fig. 3. To prevent infinite runs the procedure terminates if the maximal number of restarts is reached.

3. EXPERIMENTAL PROCEDURE

The gain rates were set to their recommended values `a0` = 0.602 and `c0` = 0.101, instead of the respective optimal values. All other parameters were set as recommended in [6]. The experiments were conducted on a Cluster with 2.44 GHz CPUs (machine_type x86_64) under Octave 3.0.2.

4. RESULTS

Results from experiments according to [2] on the benchmarks functions given in [1, 3] are presented in Figures 4 and 5 and in Tables 1 and 2.

5. CPU TIMING EXPERIMENT

For the timing experiment the same multistart algorithm was run on f_8 and restarted until at least 30 seconds had passed (according to Figure 2 in [2]). The results were $1.2; 1.2; 1.2; 1.2; 1.3$ and 1.4×10^{-4} seconds per function evaluation in dimension $2; 3; 5; 10; 20$ and 40, respectively. The dependency of CPU time on the search space dimensionality is small.

6. CONCLUSION

This paper reports the result for the basic SPSA on the BBOB 2009 noisy testbed.

Acknowledgments

The author would like to acknowledge the great work of the BBOB team with particular kudos the Anne Auger, Nikolaus Hansen and Raymond Ros. This work was supported by the Austrian Science Fund (FWF) under grant P19069-N18.

GECCO'09, July 8–12, 2009, Montréal Québec, Canada.

Figure 1: SPSA in Matlab

```
% simple spsa function
function [x,termvalue] = alg(FUN, x, parameter, maxGenerations, ftarget,...
                              DIM, maxfunevals)

    % intialze counters
    k = 1;

    % initialize algorithm parameter
    a0 = parameter(1);
    alpha = parameter(2);
    c0 = parameter(3);
    gamma = parameter(4);
    A = parameter(5);
    lambda = parameter(6);

    while 1

        % gain sequences ak and ck
        ak = a0 * (A + k)^(-alpha);
        ck = c0 * k^(-gamma);

        % gradient approximation with averaging of several approximations
        delta = 2*round(rand(DIM,lambda))-1;
        X = repmat(x,1,lambda);
        yplus = FUN(X + ck.*delta);
        yminus = FUN(X - ck.*delta);
        Gk = mean(repmat((yplus-yminus),DIM,1)./(2*ck.*delta),2);

        % update objectVector
        x = x - ak*Gk;

        % termination criterions
        fit = FUN(x);
        % stop if target or maxfunevals is reached
        if fit <= ftarget ||  feval(FUN, 'evaluations') >= maxfunevals
            termvalue = 1;
            break;
        end
        % stop if maxGenerations or fit is larger 1e30 (probably divergent
        % run)
        if k > maxGenerations  || fit > 1e30
            termvalue = 0;
            break;
        end
        % stop if x has nan or inf entries
        if max(isnan(x)) == 1 || max(isinf(x)) == 1
            termvalue = 0;
            break;
        end

        % increase k
        k = k + 1;

    end % of while loop

end % of function
```

Δf	**f101 in 5-D**, N=15, mFE=513					**f101 in 20-D**, N=15, mFE=2026				
	#	ERT	10%	90%	RT_{succ}	#	ERT	10%	90%	RT_{succ}
10	14	2.3e2	1.7e2	2.8e2	2.3e2	13	1.1e3	8.9e2	1.3e3	9.7e2
1	9	6.0e2	5.0e2	6.9e2	3.9e2	12	1.7e3	1.5e3	2.0e3	1.5e3
1e−1	7	8.7e2	7.6e2	9.6e2	4.2e2	6	4.2e3	3.7e3	4.7e3	1.8e3
1e−3	4	1.8e3	1.7e3	1.9e3	5.1e2	4	7.0e3	6.6e3	7.3e3	2.0e3
1e−5	1	7.5e3	7.4e3	7.7e3	5.1e2	0	*28e−2*	*35e−5*	*18e+0*	2.0e3
1e−8	0	*73e−2*	*16e−6*	*89e−1*	4.5e2	.	.	.	.	.

Δf	**f102 in 5-D**, N=15, mFE=513					**f102 in 20-D**, N=15, mFE=2004				
	#	ERT	10%	90%	RT_{succ}	#	ERT	10%	90%	RT_{succ}
10	11	3.1e2	2.2e2	4.0e2	2.5e2	3	9.4e3	8.9e3	1.0e4	2.0e3
1	8	7.1e2	6.1e2	8.1e2	3.6e2	0	*48e+0*	*36e−1*	*10e+1*	2.0e3
1e−1	4	1.7e3	1.6e3	1.9e3	4.9e2	.	.	.	.	.
1e−3	2	3.8e3	3.7e3	3.8e3	5.1e2	.	.	.	.	.
1e−5	0	*84e−2*	*52e−5*	*15e+0*	4.5e2	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.

Δf	**f103 in 5-D**, N=15, mFE=513					**f103 in 20-D**, N=15, mFE=2013				
	#	ERT	10%	90%	RT_{succ}	#	ERT	10%	90%	RT_{succ}
10	6	9.7e2	8.1e2	1.1e3	4.0e2	5	4.7e3	4.1e3	5.4e3	1.7e3
1	4	1.6e3	1.4e3	1.8e3	4.1e2	3	9.1e3	8.2e3	9.8e3	1.8e3
1e−1	3	2.3e3	2.1e3	2.5e3	5.1e2	0	*17e+1*	*16e−2*	*24e+1*	8.9e1
1e−3	1	7.4e3	7.1e3	7.7e3	5.1e2	.	.	.	.	.
1e−5	0	*19e+0*	*78e−4*	*78e+0*	2.2e2	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.

Δf	**f104 in 5-D**, N=15, mFE=514					**f104 in 20-D**, N=15, mFE=2004				
	#	ERT	10%	90%	RT_{succ}	#	ERT	10%	90%	RT_{succ}
10	0	*14e+2*	*11e+1*	*38e+3*	4.5e2	0	*11e+4*	*12e+3*	*68e+4*	1.8e3
1	.	.	.	.	.	.	.	.	.	.
1e−1	.	.	.	.	.	.	.	.	.	.
1e−3	.	.	.	.	.	.	.	.	.	.
1e−5	.	.	.	.	.	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.

Δf	**f105 in 5-D**, N=15, mFE=513					**f105 in 20-D**, N=15, mFE=2004				
	#	ERT	10%	90%	RT_{succ}	#	ERT	10%	90%	RT_{succ}
10	0	*14e+2*	*42e+0*	*31e+3*	4.5e2	0	*26e+4*	*95e+3*	*60e+4*	1.8e3
1	.	.	.	.	.	.	.	.	.	.
1e−1	.	.	.	.	.	.	.	.	.	.
1e−3	.	.	.	.	.	.	.	.	.	.
1e−5	.	.	.	.	.	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.

Δf	**f106 in 5-D**, N=15, mFE=513					**f106 in 20-D**, N=15, mFE=2011				
	#	ERT	10%	90%	RT_{succ}	#	ERT	10%	90%	RT_{succ}
10	0	*17e+2*	*21e+1*	*23e+3*	5.0e2	0	*35e+2*	*83e+1*	*19e+3*	2.0e3
1	.	.	.	.	.	.	.	.	.	.
1e−1	.	.	.	.	.	.	.	.	.	.
1e−3	.	.	.	.	.	.	.	.	.	.
1e−5	.	.	.	.	.	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.

Δf	**f107 in 5-D**, N=15, mFE=513					**f107 in 20-D**, N=15, mFE=2004				
	#	ERT	10%	90%	RT_{succ}	#	ERT	10%	90%	RT_{succ}
10	2	3.6e3	3.4e3	3.8e3	2.9e2	0	*13e+1*	*54e+0*	*22e+1*	8.9e2
1	0	*30e+0*	*65e−1*	*48e+0*	3.5e2	.	.	.	.	.
1e−1	.	.	.	.	.	.	.	.	.	.
1e−3	.	.	.	.	.	.	.	.	.	.
1e−5	.	.	.	.	.	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.

Δf	**f108 in 5-D**, N=15, mFE=513					**f108 in 20-D**, N=15, mFE=2004				
	#	ERT	10%	90%	RT_{succ}	#	ERT	10%	90%	RT_{succ}
10	1	7.4e3	7.2e3	7.7e3	5.1e2	0	*21e+1*	*15e+1*	*33e+1*	5.0e1
1	0	*47e+0*	*24e+0*	*72e+0*	2.2e2	.	.	.	.	.
1e−1	.	.	.	.	.	.	.	.	.	.
1e−3	.	.	.	.	.	.	.	.	.	.
1e−5	.	.	.	.	.	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.

Δf	**f109 in 5-D**, N=15, mFE=513					**f109 in 20-D**, N=15, mFE=2019				
	#	ERT	10%	90%	RT_{succ}	#	ERT	10%	90%	RT_{succ}
10	8	6.2e2	5.1e2	7.3e2	2.2e2	9	2.0e3	1.6e3	2.4e3	1.2e3
1	5	1.3e3	1.2e3	1.4e3	4.5e2	5	5.2e3	4.6e3	5.7e3	2.0e3
1e−1	2	3.8e3	3.8e3	3.8e3	5.1e2	1	2.9e4	2.8e4	3.0e4	2.0e3
1e−3	0	*69e−1*	*92e−3*	*40e+0*	4.5e2	0	*18e−1*	*18e−2*	*25e+1*	1.3e3
1e−5	.	.	.	.	.	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.

Δf	**f110 in 5-D**, N=15, mFE=513					**f110 in 20-D**, N=15, mFE=2004				
	#	ERT	10%	90%	RT_{succ}	#	ERT	10%	90%	RT_{succ}
10	0	*22e+3*	*45e+2*	*63e+3*	1.2e2	0	*31e+4*	*20e+4*	*52e+4*	1.0e0
1	.	.	.	.	.	.	.	.	.	.
1e−1	.	.	.	.	.	.	.	.	.	.
1e−3	.	.	.	.	.	.	.	.	.	.
1e−5	.	.	.	.	.	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.

Δf	**f111 in 5-D**, N=15, mFE=513					**f111 in 20-D**, N=15, mFE=2004				
	#	ERT	10%	90%	RT_{succ}	#	ERT	10%	90%	RT_{succ}
10	0	*43e+3*	*11e+3*	*15e+4*	8.9e1	0	*29e+4*	*18e+4*	*65e+4*	8.9e1
1	.	.	.	.	.	.	.	.	.	.
1e−1	.	.	.	.	.	.	.	.	.	.
1e−3	.	.	.	.	.	.	.	.	.	.
1e−5	.	.	.	.	.	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.

Δf	**f112 in 5-D**, N=15, mFE=513					**f112 in 20-D**, N=15, mFE=2016				
	#	ERT	10%	90%	RT_{succ}	#	ERT	10%	90%	RT_{succ}
10	0	*85e+1*	*76e+0*	*69e+2*	4.5e2	0	*39e+2*	*27e+1*	*17e+3*	1.8e3
1	.	.	.	.	.	.	.	.	.	.
1e−1	.	.	.	.	.	.	.	.	.	.
1e−3	.	.	.	.	.	.	.	.	.	.
1e−5	.	.	.	.	.	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.

Δf	**f113 in 5-D**, N=15, mFE=513					**f113 in 20-D**, N=15, mFE=2004				
	#	ERT	10%	90%	RT_{succ}	#	ERT	10%	90%	RT_{succ}
10	1	7.6e3	7.5e3	7.7e3	5.1e2	0	*22e+2*	*67e+1*	*40e+2*	5.0e1
1	0	*29e+1*	*21e+0*	*12e+2*	3.2e2	.	.	.	.	.
1e−1	.	.	.	.	.	.	.	.	.	.
1e−3	.	.	.	.	.	.	.	.	.	.
1e−5	.	.	.	.	.	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.

Δf	**f114 in 5-D**, N=15, mFE=513					**f114 in 20-D**, N=15, mFE=2004				
	#	ERT	10%	90%	RT_{succ}	#	ERT	10%	90%	RT_{succ}
10	0	*19e+1*	*32e+0*	*88e+1*	7.0e1	0	*15e+2*	*94e+1*	*34e+2*	7.0e1
1	.	.	.	.	.	.	.	.	.	.
1e−1	.	.	.	.	.	.	.	.	.	.
1e−3	.	.	.	.	.	.	.	.	.	.
1e−5	.	.	.	.	.	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.

Δf	**f115 in 5-D**, N=15, mFE=528					**f115 in 20-D**, N=15, mFE=2040				
	#	ERT	10%	90%	RT_{succ}	#	ERT	10%	90%	RT_{succ}
10	3	2.4e3	2.2e3	2.6e3	5.1e2	2	1.4e4	1.3e4	1.5e4	1.3e3
1	0	*30e+0*	*63e−1*	*17e+1*	4.0e2	0	*63e+0*	*41e−1*	*20e+2*	1.8e3
1e−1	.	.	.	.	.	.	.	.	.	.
1e−3	.	.	.	.	.	.	.	.	.	.
1e−5	.	.	.	.	.	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.

Δf	**f116 in 5-D**, N=15, mFE=513					**f116 in 20-D**, N=15, mFE=2004				
	#	ERT	10%	90%	RT_{succ}	#	ERT	10%	90%	RT_{succ}
10	0	*33e+3*	*42e+2*	*24e+4*	7.0e1	0	*12e+4*	*72e+3*	*29e+4*	5.0e1
1	.	.	.	.	.	.	.	.	.	.
1e−1	.	.	.	.	.	.	.	.	.	.
1e−3	.	.	.	.	.	.	.	.	.	.
1e−5	.	.	.	.	.	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.

Δf	**f117 in 5-D**, N=15, mFE=513					**f117 in 20-D**, N=15, mFE=2004				
	#	ERT	10%	90%	RT_{succ}	#	ERT	10%	90%	RT_{succ}
10	0	*27e+3*	*12e+2*	*12e+4*	1.1e2	0	*16e+4*	*73e+3*	*32e+4*	1.0e0
1	.	.	.	.	.	.	.	.	.	.
1e−1	.	.	.	.	.	.	.	.	.	.
1e−3	.	.	.	.	.	.	.	.	.	.
1e−5	.	.	.	.	.	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.

Δf	**f118 in 5-D**, N=15, mFE=513					**f118 in 20-D**, N=15, mFE=2004				
	#	ERT	10%	90%	RT_{succ}	#	ERT	10%	90%	RT_{succ}
10	0	*96e+1*	*22e+1*	*10e+3*	4.5e2	0	*47e+2*	*12e+2*	*13e+4*	1.8e3
1	.	.	.	.	.	.	.	.	.	.
1e−1	.	.	.	.	.	.	.	.	.	.
1e−3	.	.	.	.	.	.	.	.	.	.
1e−5	.	.	.	.	.	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.

Δf	**f119 in 5-D**, N=15, mFE=519					**f119 in 20-D**, N=15, mFE=2004				
	#	ERT	10%	90%	RT_{succ}	#	ERT	10%	90%	RT_{succ}
10	9	4.7e2	3.6e2	5.9e2	2.5e2	0	*13e+1*	*53e+0*	*31e+1*	5.0e1
1	0	*74e−1*	*36e−1*	*26e+0*	3.2e2	.	.	.	.	.
1e−1	.	.	.	.	.	.	.	.	.	.
1e−3	.	.	.	.	.	.	.	.	.	.
1e−5	.	.	.	.	.	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.

Δf	**f120 in 5-D**, N=15, mFE=513					**f120 in 20-D**, N=15, mFE=2004				
	#	ERT	10%	90%	RT_{succ}	#	ERT	10%	90%	RT_{succ}
10	4	1.6e3	1.4e3	1.8e3	3.8e2	0	*83e+0*	*36e+0*	*22e+1*	1.0e0
1	0	*25e+0*	*36e−1*	*14e+1*	1.4e2	.	.	.	.	.
1e−1	.	.	.	.	.	.	.	.	.	.
1e−3	.	.	.	.	.	.	.	.	.	.
1e−5	.	.	.	.	.	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.

Table 1: Shown are, for functions f_{101}-f_{120} and for a given target difference to the optimal function value Δf: the number of successful trials (#); the expected running time to surpass $f_{\mathrm{opt}} + \Delta f$ (ERT, see Figure 4); the 10%-tile and 90%-tile of the bootstrap distribution of ERT; the average number of function evaluations in successful trials or, if none was successful, as last entry the median number of function evaluations to reach the best function value ($\mathbf{RT_{succ}}$). If $f_{\mathrm{opt}} + \Delta f$ was never reached, figures in *italics* denote the best achieved Δf-value of the median trial and the 10% and 90%-tile trial. Furthermore, N denotes the number of trials, and mFE denotes the maximum of number of function evaluations executed in one trial. See Figure 4 for the names of functions.

Figure 2: Determination of the initial values for the gains

```
function [a0,c0] = DetermineParameter(A,alpha,x,step,FUN,DIM)

    % generate matrix with trial vectors
    X = repmat(x,1,10);

    % c0
    dummy = FUN(X);
    c0 = max([std(dummy,1),1e-5]);

    % a0
    % generation of the simultaneous perturbation vector
    delta = 2*round(rand(DIM,10))-1;

    % function evaluation
    yplus = FUN(X + c0.*delta);
    yminus = FUN(X - c0.*delta);

    % gradient approximation
    gApprox = mean(repmat((yplus-yminus),DIM,1)./(2*c0.*delta),2);

    % mean of the magnitude of gradient element
    gMeanElement = abs(mean(gApprox));

    % determine parameter a
    a0 = step*(1+A)^alpha/gMeanElement;

end % of function
```

Δf	**f121 in 5-D**, N=15, mFE=513					**f121 in 20-D**, N=15, mFE=2044				
	#	ERT	10%	90%	RT_{succ}	#	ERT	10%	90%	RT_{succ}
10	11	2.5e2	1.6e2	3.5e2	1.7e2	6	4.0e3	3.5e3	4.4e3	1.5e3
1	2	3.6e3	3.4e3	3.8e3	5.1e2	0	*13e+0*	*52e−1*	*41e+0*	1.8e3
1e−1	1	7.6e3	7.6e3	7.7e3	5.1e2	.	.	.	.	.
1e−3	0	*42e−1*	*45e−2*	*35e+0*	4.5e2	.	.	.	.	.
1e−5	.	.	.	.	.	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.

Δf	**f122 in 5-D**, N=15, mFE=513					**f122 in 20-D**, N=15, mFE=2023				
	#	ERT	10%	90%	RT_{succ}	#	ERT	10%	90%	RT_{succ}
10	4	1.5e3	1.2e3	1.7e3	5.1e2	0	*32e+0*	*14e+0*	*16e+1*	1.0e0
1	0	*13e+0*	*43e−1*	*25e+0*	1.1e2	.	.	.	.	.
1e−1	.	.	.	.	.	.	.	.	.	.
1e−3	.	.	.	.	.	.	.	.	.	.
1e−5	.	.	.	.	.	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.

Δf	**f123 in 5-D**, N=15, mFE=579					**f123 in 20-D**, N=15, mFE=2181				
	#	ERT	10%	90%	RT_{succ}	#	ERT	10%	90%	RT_{succ}
10	3	2.2e3	1.9e3	2.5e3	3.8e2	0	*23e+0*	*14e+0*	*38e+0*	1.0e3
1	0	*13e+0*	*72e−1*	*42e+0*	1.0e2	.	.	.	.	.
1e−1	.	.	.	.	.	.	.	.	.	.
1e−3	.	.	.	.	.	.	.	.	.	.
1e−5	.	.	.	.	.	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.

Δf	**f124 in 5-D**, N=15, mFE=531					**f124 in 20-D**, N=15, mFE=2065				
	#	ERT	10%	90%	RT_{succ}	#	ERT	10%	90%	RT_{succ}
10	9	4.2e2	3.0e2	5.4e2	2.1e2	8	2.3e3	1.8e3	2.8e3	1.3e3
1	0	*78e−1*	*13e−1*	*31e+0*	3.5e2	0	*97e−1*	*62e−1*	*14e+0*	1.3e3
1e−1	.	.	.	.	.	.	.	.	.	.
1e−3	.	.	.	.	.	.	.	.	.	.
1e−5	.	.	.	.	.	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.

Δf	**f125 in 5-D**, N=15, mFE=522					**f125 in 20-D**, N=15, mFE=2013				
	#	ERT	10%	90%	RT_{succ}	#	ERT	10%	90%	RT_{succ}
10	15	1.0e0	1.0e0	1.0e0	1.0e0	15	1.0e0	1.0e0	1.0e0	1.0e0
1	15	1.1e2	7.9e1	1.4e2	1.1e2	11	1.1e3	7.8e2	1.5e3	8.8e2
1e−1	0	*42e−2*	*17e−2*	*66e−2*	2.8e2	0	*77e−2*	*63e−2*	*13e−1*	1.3e3
1e−3	.	.	.	.	.	.	.	.	.	.
1e−5	.	.	.	.	.	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.

Δf	**f126 in 5-D**, N=15, mFE=513					**f126 in 20-D**, N=15, mFE=2004				
	#	ERT	10%	90%	RT_{succ}	#	ERT	10%	90%	RT_{succ}
10	15	1.0e0	1.0e0	1.0e0	1.0e0	15	1.0e0	1.0e0	1.0e0	1.0e0
1	4	1.5e3	1.3e3	1.7e3	4.0e2	0	*25e−1*	*19e−1*	*37e−1*	5.0e1
1e−1	0	*19e−1*	*35e−2*	*44e−1*	8.9e1	.	.	.	.	.
1e−3	.	.	.	.	.	.	.	.	.	.
1e−5	.	.	.	.	.	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.

Δf	**f127 in 5-D**, N=15, mFE=527					**f127 in 20-D**, N=15, mFE=2058				
	#	ERT	10%	90%	RT_{succ}	#	ERT	10%	90%	RT_{succ}
10	15	1.0e0	1.0e0	1.0e0	1.0e0	15	1.0e0	1.0e0	1.0e0	1.0e0
1	13	1.5e2	8.4e1	2.1e2	1.4e2	8	2.3e3	1.8e3	2.9e3	9.4e2
1e−1	0	*37e−2*	*23e−2*	*15e−1*	1.2e2	0	*97e−2*	*61e−2*	*18e−1*	6.3e2
1e−3	.	.	.	.	.	.	.	.	.	.
1e−5	.	.	.	.	.	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.

Δf	**f128 in 5-D**, N=15, mFE=513					**f128 in 20-D**, N=15, mFE=2013				
	#	ERT	10%	90%	RT_{succ}	#	ERT	10%	90%	RT_{succ}
10	1	7.5e3	7.4e3	7.7e3	5.1e2	0	*76e+0*	*74e+0*	*80e+0*	2.8e2
1	0	*39e+0*	*14e+0*	*60e+0*	1.8e2	.	.	.	.	.
1e−1	.	.	.	.	.	.	.	.	.	.
1e−3	.	.	.	.	.	.	.	.	.	.
1e−5	.	.	.	.	.	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.

Δf	**f129 in 5-D**, N=15, mFE=513					**f129 in 20-D**, N=15, mFE=2004				
	#	ERT	10%	90%	RT_{succ}	#	ERT	10%	90%	RT_{succ}
10	1	7.2e3	6.6e3	7.7e3	5.1e2	0	*78e+0*	*74e+0*	*81e+0*	3.2e2
1	0	*35e+0*	*15e+0*	*55e+0*	1.4e2	.	.	.	.	.
1e−1	.	.	.	.	.	.	.	.	.	.
1e−3	.	.	.	.	.	.	.	.	.	.
1e−5	.	.	.	.	.	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.

Δf	**f130 in 5-D**, N=15, mFE=513					**f130 in 20-D**, N=15, mFE=2046				
	#	ERT	10%	90%	RT_{succ}	#	ERT	10%	90%	RT_{succ}
10	7	7.2e2	5.8e2	8.6e2	3.1e2	0	*77e+0*	*20e+0*	*84e+0*	3.5e2
1	0	*14e+0*	*20e−1*	*42e+0*	2.2e2	.	.	.	.	.
1e−1	.	.	.	.	.	.	.	.	.	.
1e−3	.	.	.	.	.	.	.	.	.	.
1e−5	.	.	.	.	.	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.

Table 2: Shown are, for functions f_{121}-f_{130} and for a given target difference to the optimal function value Δf: the number of successful trials (#); the expected running time to surpass $f_{\mathrm{opt}} + \Delta f$ (ERT, see Figure 4); the 10%-tile and 90%-tile of the bootstrap distribution of ERT; the average number of function evaluations in successful trials or, if none was successful, as last entry the median number of function evaluations to reach the best function value ($\mathbf{RT_{succ}}$). If $f_{\mathrm{opt}} + \Delta f$ was never reached, figures in *italics* denote the best achieved Δf-value of the median trial and the 10% and 90%-tile trial. Furthermore, N denotes the number of trials, and mFE denotes the maximum of number of function evaluations executed in one trial. See Figure 4 for the names of functions.

Figure 3: Restart procedure for the SPSA

```
function x = spsa(FUN, DIM, ftarget, maxfunevals)

    % make sure to terminate
    if isinf(maxfunevals) || maxfunevals > 1e5*DIM
        kmax = 1e5*DIM;
    else
        kmax = maxfunevals;
    end

    % constant parameter
    A = 0.1*kmax;   % A approx 10% of max generations
    gamma = 0.101;  % reduction rate for ck (as recommended)
    alpha = 0.602;  % reduction rate for ak (as recommended)

    % multistart such that ftarget is reached with reasonable prob.
    for ilaunch = 1:100  % relaunch optimizer up to 100 times

        % restarts
        if ilaunch == 1                    % initial scenario
           xstart = 8 * rand(DIM, 1) - 4;  % random start solution
           step = 0.1;                     % parameter to determine a0
           lambda = 10;                    % number of gradient approximations
           [a0,c0] = DetermineParameter(A,alpha,xstart,step,FUN,DIM);

        else

            choice = round(3*rand) + 1;

            % if the xstart is changing, parameter a0 has to be newly
            % calcualeted

            switch choice

                case 1     % new point
                    xstart = 8 * rand(DIM, 1) - 4;
                    [a0,c0] = DetermineParameter(A,alpha,xstart,step,FUN,DIM);

                case 2     % improve old point
                    xstart = x;
                    [a0,c0] = DetermineParameter(A,alpha,xstart,step,FUN,DIM);

                case 3     % half the step size
                    step = step/2;
                    [a0,c0] = DetermineParameter(A,alpha,xstart,step,FUN,DIM);

                case 4     % increase lambda
                    lambda = ceil(lambda * sqrt(2));

            end % switch case

        end

        % try spsa
        parameter = [a0,alpha,c0,gamma,A,lambda];
        [x,termvalue] = alg(FUN,xstart,parameter,kmax,ftarget,DIM,maxfunevals);

        if termvalue == 1
            break;
        end

    end

end % of function
```

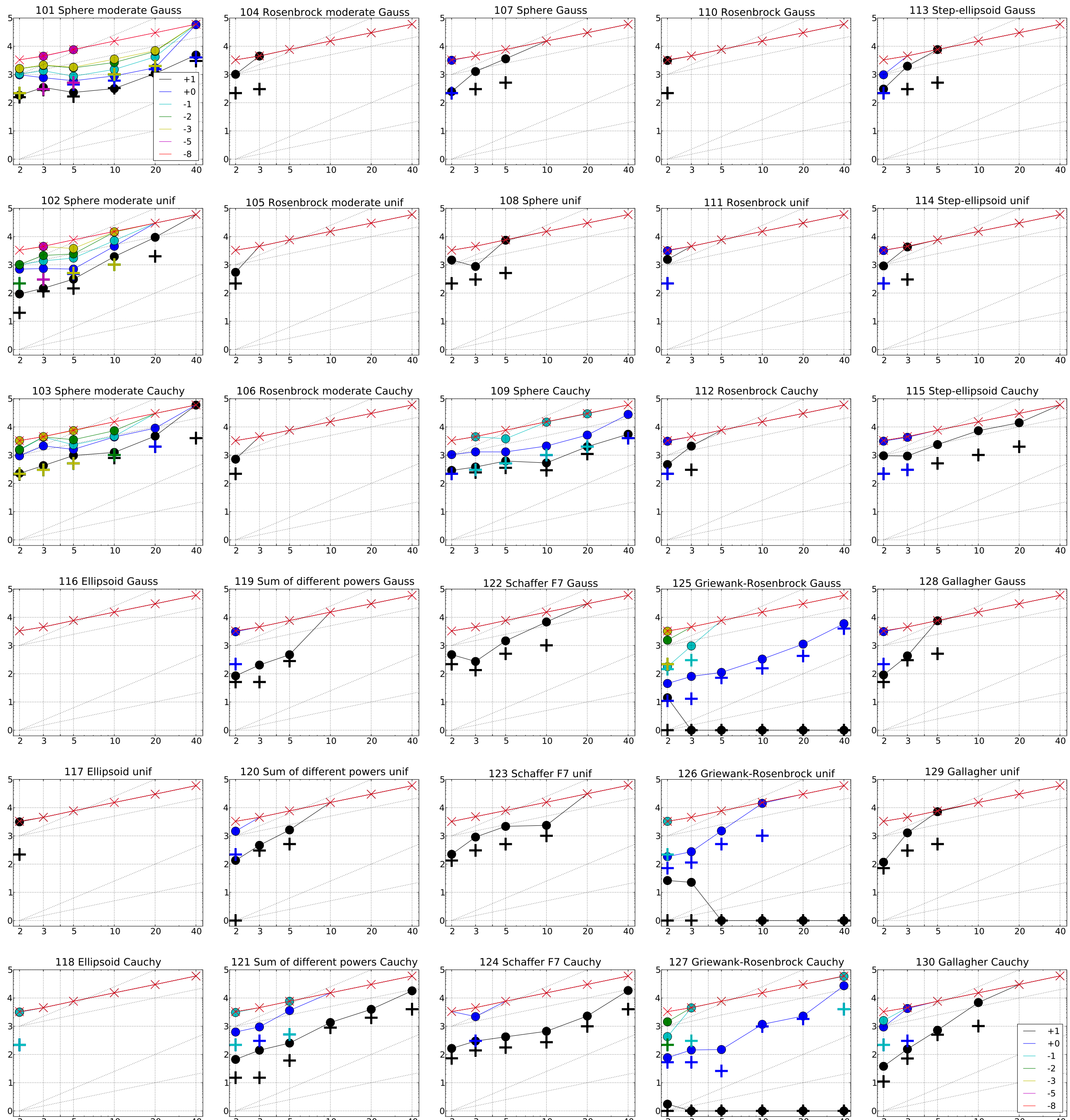

Figure 4: Expected Running Time (ERT, ●) to reach $f_{\mathrm{opt}} + \Delta f$ and median number of function evaluations of successful trials (+), shown for $\Delta f = 10, 1, 10^{-1}, 10^{-2}, 10^{-3}, 10^{-5}, 10^{-8}$ (the exponent is given in the legend of f_{101} and f_{130}) versus dimension in log-log presentation. The $\mathrm{ERT}(\Delta f)$ equals to $\#\mathrm{FEs}(\Delta f)$ divided by the number of successful trials, where a trial is successful if $f_{\mathrm{opt}} + \Delta f$ was surpassed during the trial. The $\#\mathrm{FEs}(\Delta f)$ are the total number of function evaluations while $f_{\mathrm{opt}} + \Delta f$ was not surpassed during the trial from all respective trials (successful and unsuccessful), and f_{opt} denotes the optimal function value. Crosses ($\times$) indicate the total number of function evaluations $\#\mathrm{FEs}(-\infty)$. Numbers above ERT-symbols indicate the number of successful trials. Annotated numbers on the ordinate are decimal logarithms. Additional grid lines show linear and quadratic scaling.

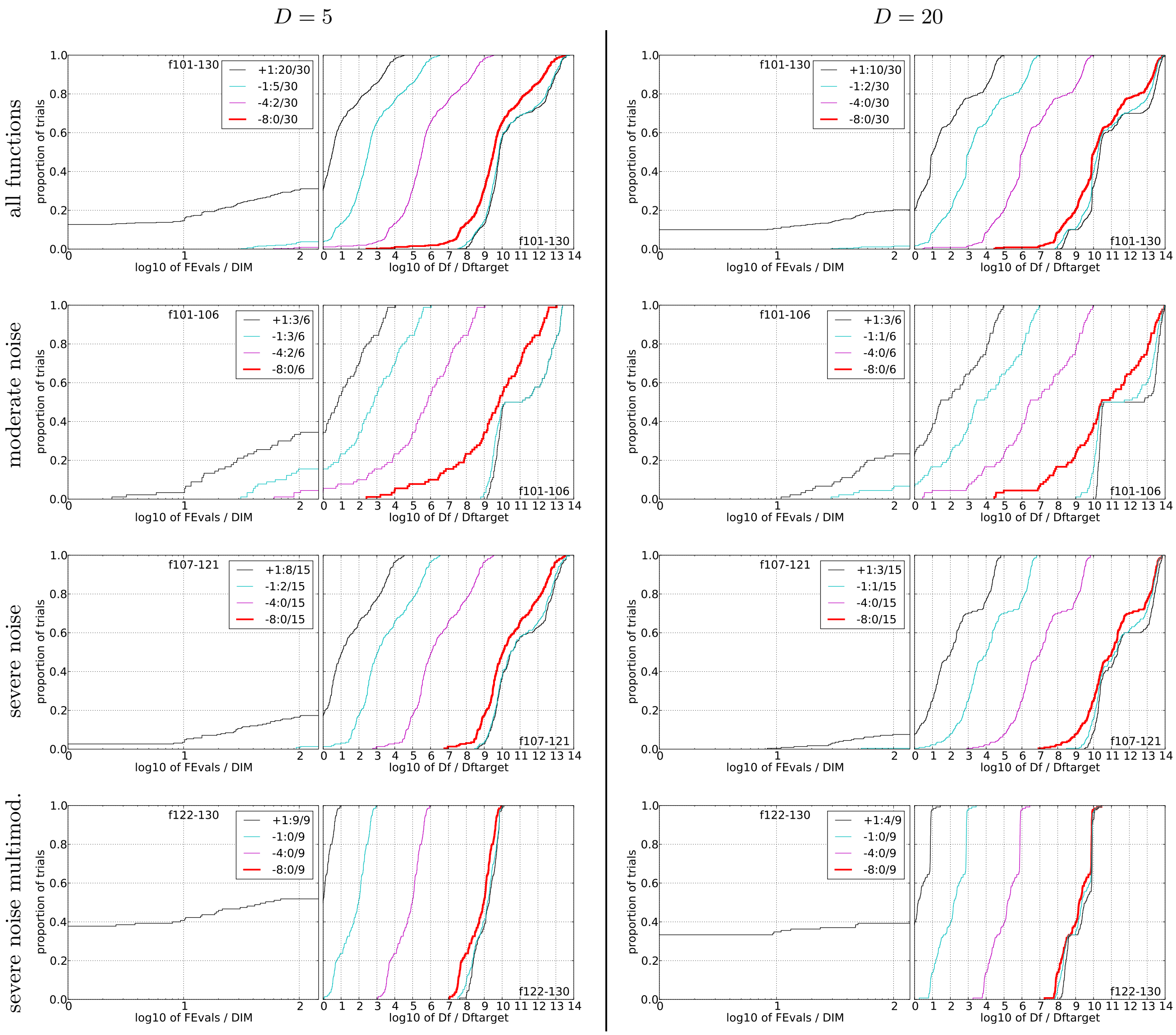

Figure 5: Empirical cumulative distribution functions (ECDFs), plotting the fraction of trials versus running time (left) or Δf. Left subplots: ECDF of the running time (number of function evaluations), divided by search space dimension D, to fall below $f_{\mathrm{opt}} + \Delta f$ with $\Delta f = 10^k$, where k is the first value in the legend. Right subplots: ECDF of the best achieved Δf divided by 10^k (upper left lines in continuation of the left subplot), and best achieved Δf divided by 10^{-8} for running times of $D, 10\,D, 100\,D \ldots$ function evaluations (from right to left cycling black-cyan-magenta). Top row: all results from all functions; second row: moderate noise functions; third row: severe noise functions; fourth row: severe noise and highly-multimodal functions. The legends indicate the number of functions that were solved in at least one trial. FEvals denotes number of function evaluations, D and DIM denote search space dimension, and Δf and Df denote the difference to the optimal function value.

7. REFERENCES

[1] S. Finck, N. Hansen, R. Ros, and A. Auger. Real-parameter black-box optimization benchmarking 2009: Presentation of the noisy functions. Technical Report 2009/21, Research Center PPE, 2009.

[2] N. Hansen, A. Auger, S. Finck, and R. Ros. Real-parameter black-box optimization benchmarking 2009: Experimental setup. Technical Report RR-6828, INRIA, 2009.

[3] N. Hansen, S. Finck, R. Ros, and A. Auger. Real-parameter black-box optimization benchmarking 2009: Noisy functions definitions. Technical Report RR-6869, INRIA, 2009.

[4] J. C. Spall. Multivariate Stochastic Approximation Using a Simultaneous Perturbation Gradient Approximation. *IEEE Transactions on Automatic Control*, 37(3):332–341, March 1992.

[5] J. C. Spall. An Overview of the Simultaneous Perturbation Method for Efficient Optimization. *Johns Hopkins APL Technical Digest*, 19:482–492, 1998.

[6] J. C. Spall. *Introduction to Stochastic Search and Optimization*. John Wiley & Sons, Hoboken, NJ, 2003.