Black-Box Optimization Benchmarking Template for Noisy Function Testbed

An Example BBOB 2009 Workshop Paper

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

The BBOBies

ABSTRACT

As an example, we benchmark the Simultaneous Perturbation Stochastic Algorithm (SPS) algorithm on the noisy BBOB 2009 testbed. Each canditate solution is sampled uniformly in $[-5,5]^{\mathrm{DIM}}$, where DIM represents the search space dimension. The maximal number of function evaluation is set to 1000 times DIM.

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

1. INTRODUCTION

The SPSA method [4] is a real-parameter black-box optimization method that approximates the gradient at the current search point with just $2 \times \lambda$ function evaluations, independent of the search space dimension. The parameter λ represents the number of approximations used to determine the gradient. In this paper, a multistart version of the SPSA method is benchmarked on the noisy BBOB 2009 testbed [1, 3] according to the experimental design from [2], cf. to Figure 1.

2. ALGORITHM PRESENTATION

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

The 3 parts of the SPSA with restarts are shown in Fig. 1 – Fig. 3. In Fig. 1 the main function is displayed. In Fig. 2 the used multi start procedure is shown. Finally, in Fig. 3 the determination of the initial step sizes a_0 (for the update of the search point) and c_0 (for the test step) is presented. The step size reductions are chosen as recommended by Spall in [5].

3. RESULTS AND DISCUSSION

The results are presented in Tables 1 and 2 and in Figures 4 and 5. The algorithm solves at least one trial of f101 for DIM $= (2,3,5)$, f102 for DIM $= (2,3)$, and DIM $= 3$ for f103. The reason for the low convergence rate is the use of a relatively small number of maximal function evaluations $1e3 \times \mathrm{DIM}$.

4. 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]). These experiments have been conducted with an Intel Pentium 4 CPU processor with 3.0 GHz under Linux 2.6.27-19-3.2-pae using Octave 3.0.2 The results were $1.9; 2.0; 2.0; 1.5; 1.5$ and 1.9×10^{-3} seconds per function evaluation in dimension $2; 3; 5; 10; 20$ and 40, respectively. Up to 40-D the dependency of CPU time on the search space dimensionality is small.

5. CONCLUSION

The SPSA algorithm, as implemented in Octave 3.0.2, equipped with a multistart procedure, is a common and reliable black-box search algorithm for noisy benchmark functions. In the given settings (low number of maximal function evaluations) the algorithm is not able to solve one benchmark function for all dimensions.

Acknowledgments

The author would like to acknowledge the great and hard work of the BBOB team with particular kudos to Anne Auger, Nikolaus Hansen and Raymond Ros.

6. REFERENCES

[1] S. Finck, N. Hansen, R. Ros, and A. Auger. Real-parameter black-box optimization benchmarking 2009: Presentation of the noiseless functions. Technical Report 2009/20, Research Center PPE, 2009.

	***f*101 in 5-D**, N=15, mFE=5014					***f*101 in 20-D**, N=15, mFE=20014				
Δf	#	ERT	10%	90%	RT_{succ}	#	ERT	10%	90%	RT_{succ}
10	13	8.3e2	8.9e1	1.6e3	4.4e2	2	1.2e5	1.1e5	1.4e5	1.0e4
1	13	9.1e2	2.0e2	1.7e3	5.2e2	2	1.2e5	1.1e5	1.4e5	1.0e4
1e−1	13	1.0e3	3.5e2	1.7e3	6.2e2	2	1.2e5	1.1e5	1.4e5	1.0e4
1e−3	11	2.2e3	1.3e3	3.1e3	1.2e3	2	1.2e5	1.1e5	1.4e5	1.1e4
1e−5	9	3.9e3	2.7e3	5.2e3	2.1e3	0	*11e+1*	*68e−6*	*13e+1*	1.0e4
1e−8	3	2.1e4	1.8e4	2.3e4	3.5e3	.	.	.	.	.

	***f*102 in 5-D**, N=15, mFE=5006					***f*102 in 20-D**, N=15, mFE=20038				
Δf	#	ERT	10%	90%	RT_{succ}	#	ERT	10%	90%	RT_{succ}
10	12	1.3e3	5.1e2	2.2e3	1.3e3	2	1.3e5	1.1e5	1.4e5	1.1e4
1	12	2.1e3	1.2e3	3.0e3	1.4e3	0	*87e+0*	*46e−1*	*16e+1*	4.5e3
1e−1	10	3.3e3	2.2e3	4.3e3	1.4e3	.	.	.	.	.
1e−3	7	7.1e3	5.6e3	8.6e3	2.0e3	.	.	.	.	.
1e−5	2	3.5e4	3.2e4	3.8e4	2.6e3	.	.	.	.	.
1e−8	0	*33e−4*	*36e−7*	*17e+0*	1.3e3	.	.	.	.	.

	***f*103 in 5-D**, N=15, mFE=5003					***f*103 in 20-D**, N=15, mFE=20005				
Δf	#	ERT	10%	90%	RT_{succ}	#	ERT	10%	90%	RT_{succ}
10	7	5.8e3	3.7e3	7.9e3	3.6e3	3	8.4e4	7.2e4	9.5e4	8.5e3
1	6	7.6e3	5.9e3	9.4e3	4.2e3	2	1.4e5	1.2e5	1.5e5	1.3e4
1e−1	6	7.7e3	6.0e3	9.5e3	4.2e3	2	1.4e5	1.2e5	1.5e5	1.3e4
1e−3	3	2.0e4	1.7e4	2.3e4	5.0e3	1	2.9e5	2.7e5	3.0e5	6.7e3
1e−5	1	7.1e4	6.6e4	7.5e4	5.0e3	0	*20e+1*	*17e−3*	*28e+1*	1.1e2
1e−8	0	*15e+0*	*54e−6*	*84e+0*	2.8e2	.	.	.	.	.

	***f*104 in 5-D**, N=15, mFE=5007					***f*104 in 20-D**, N=15, mFE=20005				
Δf	#	ERT	10%	90%	RT_{succ}	#	ERT	10%	90%	RT_{succ}
10	0	*35e+1*	*40e+0*	*20e+3*	3.5e2	0	*88e+3*	*38e+2*	*78e+4*	4.5e3
1	.	.	.	.	.	.	.	.	.	.
1e−1	.	.	.	.	.	.	.	.	.	.
1e−3	.	.	.	.	.	.	.	.	.	.
1e−5	.	.	.	.	.	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.

	***f*105 in 5-D**, N=15, mFE=5007					***f*105 in 20-D**, N=15, mFE=20005				
Δf	#	ERT	10%	90%	RT_{succ}	#	ERT	10%	90%	RT_{succ}
10	0	*84e+1*	*66e+0*	*53e+3*	1.1e3	0	*11e+4*	*10e+3*	*77e+4*	4.5e3
1	.	.	.	.	.	.	.	.	.	.
1e−1	.	.	.	.	.	.	.	.	.	.
1e−3	.	.	.	.	.	.	.	.	.	.
1e−5	.	.	.	.	.	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.

	***f*106 in 5-D**, N=15, mFE=5002					***f*106 in 20-D**, N=15, mFE=20003				
Δf	#	ERT	10%	90%	RT_{succ}	#	ERT	10%	90%	RT_{succ}
10	1	7.1e4	6.6e4	7.5e4	5.0e3	0	*55e+1*	*17e+1*	*43e+4*	5.0e3
1	0	*18e+1*	*21e+0*	*92e+2*	1.3e3	.	.	.	.	.
1e−1	.	.	.	.	.	.	.	.	.	.
1e−3	.	.	.	.	.	.	.	.	.	.
1e−5	.	.	.	.	.	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.

	***f*107 in 5-D**, N=15, mFE=5029					***f*107 in 20-D**, N=15, mFE=20005				
Δf	#	ERT	10%	90%	RT_{succ}	#	ERT	10%	90%	RT_{succ}
10	3	2.1e4	1.8e4	2.3e4	6.6e2	0	*17e+1*	*59e+0*	*22e+1*	4.0e3
1	0	*26e+0*	*56e−1*	*40e+0*	4.0e2	.	.	.	.	.
1e−1	.	.	.	.	.	.	.	.	.	.
1e−3	.	.	.	.	.	.	.	.	.	.
1e−5	.	.	.	.	.	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.

	***f*108 in 5-D**, N=15, mFE=5004					***f*108 in 20-D**, N=15, mFE=20006				
Δf	#	ERT	10%	90%	RT_{succ}	#	ERT	10%	90%	RT_{succ}
10	0	*39e+0*	*18e+0*	*71e+0*	5.6e1	0	*22e+1*	*12e+1*	*28e+1*	1.0e0
1	.	.	.	.	.	.	.	.	.	.
1e−1	.	.	.	.	.	.	.	.	.	.
1e−3	.	.	.	.	.	.	.	.	.	.
1e−5	.	.	.	.	.	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.

	***f*109 in 5-D**, N=15, mFE=5003					***f*109 in 20-D**, N=15, mFE=20005				
Δf	#	ERT	10%	90%	RT_{succ}	#	ERT	10%	90%	RT_{succ}
10	9	3.8e3	2.4e3	5.1e3	3.1e3	1	2.8e5	2.6e5	3.0e5	2.0e4
1	6	7.9e3	6.1e3	9.5e3	4.2e3	1	2.8e5	2.7e5	3.0e5	2.0e4
1e−1	3	2.0e4	1.7e4	2.3e4	3.4e3	0	*13e+1*	*65e+0*	*19e+1*	5.0e3
1e−3	0	*62e−1*	*64e−4*	*90e+0*	1.3e3	.	.	.	.	.
1e−5	.	.	.	.	.	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.

	***f*110 in 5-D**, N=15, mFE=5007					***f*110 in 20-D**, N=15, mFE=20027				
Δf	#	ERT	10%	90%	RT_{succ}	#	ERT	10%	90%	RT_{succ}
10	0	*91e+2*	*72e+0*	*82e+3*	2.0e2	0	*32e+4*	*91e+3*	*56e+4*	6.3e1
1	.	.	.	.	.	.	.	.	.	.
1e−1	.	.	.	.	.	.	.	.	.	.
1e−3	.	.	.	.	.	.	.	.	.	.
1e−5	.	.	.	.	.	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.

	***f*111 in 5-D**, N=15, mFE=5004					***f*111 in 20-D**, N=15, mFE=20003				
Δf	#	ERT	10%	90%	RT_{succ}	#	ERT	10%	90%	RT_{succ}
10	0	*33e+3*	*48e+2*	*99e+3*	1.1e2	0	*24e+4*	*14e+4*	*63e+4*	1.0e0
1	.	.	.	.	.	.	.	.	.	.
1e−1	.	.	.	.	.	.	.	.	.	.
1e−3	.	.	.	.	.	.	.	.	.	.
1e−5	.	.	.	.	.	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.

	***f*112 in 5-D**, N=15, mFE=5004					***f*112 in 20-D**, N=15, mFE=20003				
Δf	#	ERT	10%	90%	RT_{succ}	#	ERT	10%	90%	RT_{succ}
10	1	7.1e4	6.8e4	7.5e4	5.0e3	0	*10e+4*	*26e+1*	*20e+4*	5.0e3
1	0	*16e+2*	*83e+0*	*23e+3*	1.3e3	.	.	.	.	.
1e−1	.	.	.	.	.	.	.	.	.	.
1e−3	.	.	.	.	.	.	.	.	.	.
1e−5	.	.	.	.	.	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.

	***f*113 in 5-D**, N=15, mFE=5011					***f*113 in 20-D**, N=15, mFE=20013				
Δf	#	ERT	10%	90%	RT_{succ}	#	ERT	10%	90%	RT_{succ}
10	4	1.5e4	1.3e4	1.7e4	3.8e3	0	*18e+2*	*11e+2*	*39e+2*	5.6e1
1	1	7.0e4	6.6e4	7.5e4	5.0e3	.	.	.	.	.
1e−1	0	*64e+0*	*73e−1*	*37e+1*	3.2e2	.	.	.	.	.
1e−3	.	.	.	.	.	.	.	.	.	.
1e−5	.	.	.	.	.	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.

	***f*114 in 5-D**, N=15, mFE=5007					***f*114 in 20-D**, N=15, mFE=20008				
Δf	#	ERT	10%	90%	RT_{succ}	#	ERT	10%	90%	RT_{succ}
10	0	*14e+1*	*36e+0*	*77e+1*	8.9e2	0	*16e+2*	*79e+1*	*45e+2*	6.3e1
1	.	.	.	.	.	.	.	.	.	.
1e−1	.	.	.	.	.	.	.	.	.	.
1e−3	.	.	.	.	.	.	.	.	.	.
1e−5	.	.	.	.	.	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.

	***f*115 in 5-D**, N=15, mFE=5004					***f*115 in 20-D**, N=15, mFE=20003				
Δf	#	ERT	10%	90%	RT_{succ}	#	ERT	10%	90%	RT_{succ}
10	6	7.7e3	6.0e3	9.4e3	3.5e3	4	6.0e4	5.1e4	6.8e4	1.6e4
1	1	7.1e4	6.6e4	7.5e4	5.0e3	0	*66e+1*	*50e−1*	*14e+2*	5.0e3
1e−1	0	*11e+0*	*12e−1*	*32e+1*	8.9e2	.	.	.	.	.
1e−3	.	.	.	.	.	.	.	.	.	.
1e−5	.	.	.	.	.	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.

	***f*116 in 5-D**, N=15, mFE=5007					***f*116 in 20-D**, N=15, mFE=20008				
Δf	#	ERT	10%	90%	RT_{succ}	#	ERT	10%	90%	RT_{succ}
10	0	*72e+2*	*96e+1*	*52e+3*	2.0e2	0	*12e+4*	*64e+3*	*38e+4*	5.0e3
1	.	.	.	.	.	.	.	.	.	.
1e−1	.	.	.	.	.	.	.	.	.	.
1e−3	.	.	.	.	.	.	.	.	.	.
1e−5	.	.	.	.	.	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.

	***f*117 in 5-D**, N=15, mFE=5015					***f*117 in 20-D**, N=15, mFE=20006				
Δf	#	ERT	10%	90%	RT_{succ}	#	ERT	10%	90%	RT_{succ}
10	0	*10e+3*	*50e+1*	*19e+4*	2.0e2	0	*12e+4*	*74e+3*	*26e+4*	5.0e3
1	.	.	.	.	.	.	.	.	.	.
1e−1	.	.	.	.	.	.	.	.	.	.
1e−3	.	.	.	.	.	.	.	.	.	.
1e−5	.	.	.	.	.	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.

	***f*118 in 5-D**, N=15, mFE=5003					***f*118 in 20-D**, N=15, mFE=20003				
Δf	#	ERT	10%	90%	RT_{succ}	#	ERT	10%	90%	RT_{succ}
10	0	*13e+2*	*16e+1*	*15e+3*	1.3e3	0	*38e+3*	*75e+1*	*12e+4*	5.0e3
1	.	.	.	.	.	.	.	.	.	.
1e−1	.	.	.	.	.	.	.	.	.	.
1e−3	.	.	.	.	.	.	.	.	.	.
1e−5	.	.	.	.	.	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.

	***f*119 in 5-D**, N=15, mFE=5008					***f*119 in 20-D**, N=15, mFE=20008				
Δf	#	ERT	10%	90%	RT_{succ}	#	ERT	10%	90%	RT_{succ}
10	11	2.2e3	1.2e3	3.2e3	1.2e3	0	*91e+0*	*36e+0*	*23e+1*	6.3e1
1	1	6.7e4	6.0e4	7.5e4	5.0e3	.	.	.	.	.
1e−1	0	*58e−1*	*32e−1*	*39e+0*	1.1e3	.	.	.	.	.
1e−3	.	.	.	.	.	.	.	.	.	.
1e−5	.	.	.	.	.	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.

	***f*120 in 5-D**, N=15, mFE=5010					***f*120 in 20-D**, N=15, mFE=20003				
Δf	#	ERT	10%	90%	RT_{succ}	#	ERT	10%	90%	RT_{succ}
10	5	1.2e4	1.0e4	1.4e4	4.8e3	0	*11e+1*	*39e+0*	*25e+1*	6.3e1
1	0	*16e+0*	*47e−1*	*57e+0*	2.8e1	.	.	.	.	.
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 $f_{\rm opt} + \Delta f$ (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 (RT_{succ}). If $f_{\rm opt} + \Delta f$ 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 1: Core function of the SPSA in Matlab/Octave

```
% simple spsa function
function x = 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);
        if fit <= ftarget || k > maxGenerations || max(isnan(x)) == 1 || max(isinf(x)) == 1 ...
               || feval(FUN, 'evaluations') >= maxfunevals
            break;
        end

        % increase k
        k = k + 1;

    end % of while loop

end % of function
```


[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: Noiseless functions definitions. Technical Report RR-6829, 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. *Introduction to Stochastic Search and Optimization*. John Wiley & Sons, Hoboken, NJ, 2003.

Figure 2: Multistart procedure SPSA in Matlab/Octave

```
% intermediate function to accompany restarts
function [x, ilaunch] = spsa(FUN, DIM, ftarget, maxfunevals)

    % make sure to terminate
    kmax = maxfunevals/12;

    % multistart such that ftarget is reached with reasonable prob.
    for ilaunch = 1:12;  % relaunch optimizer up to 100 times

        % different restart scenarios
        % use information from previous runs? which information?
        % more deterministic rule for lambda?
        if ilaunch == 1                    % initial scenario

           xstart = 8 * rand(DIM, 1) - 4;  % new random start solution
           A = 0.1*kmax;                   % A approx 10% of max generations
           gamma = 0.101;
           alpha = 0.602;
           step = 0.1;
           lambda = 1;                     % will be steadily increased
           [a0,c0] = DetermineParameter(A,alpha,xstart,step,FUN,DIM);

        else

            choice = round(3*rand) + 1;

            switch choice

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

                case 2     % increase step
                    step = max([step * 10,1e5]);
                    [a0,c0] = DetermineParameter(A,alpha,xstart,step,FUN,DIM);

                case 3     % decrease step
                    step = min([step / 10, 1e-15]);
                    [a0,c0] = DetermineParameter(A,alpha,xstart,step,FUN,DIM);

                case 4     % increase lambda
                    lambda = lambda * 2;

            end % switch case

        end

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

        % check for convergence and function budget
        if feval(FUN, 'fbest') < ftarget || feval(FUN, 'evaluations') >= maxfunevals
            break;
        end

    end

end % of function
```

Figure 3: Function to determine the initial parameter a_0 and c_0

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

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

    % a0
    % generation of the simultaneous perturbation vector
    delta = 2*round(rand(DIM,DIM))-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 = min([step*(1+A)^alpha/gMeanElement,1e10]);

end % of function
```

	f121 in 5-D, N=15, mFE=5007					**f121 in 20-D**, N=15, mFE=20005				
Δf	#	ERT	10%	90%	RT_{succ}	#	ERT	10%	90%	RT_{succ}
10	11	2.1e3	1.2e3	3.1e3	1.2e3	0	*55e+0*	*21e+0*	*10e+1*	5.0e3
1	3	2.0e4	1.7e4	2.3e4	3.4e3	.	.	.	.	.
1e−1	2	3.3e4	2.8e4	3.8e4	5.0e3	.	.	.	.	.
1e−3	0	*38e−1*	*37e−3*	*27e+0*	7.1e2	.	.	.	.	.
1e−5	.	.	.	.	.	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.

	f122 in 5-D, N=15, mFE=5007					**f122 in 20-D**, N=15, mFE=20005				
Δf	#	ERT	10%	90%	RT_{succ}	#	ERT	10%	90%	RT_{succ}
10	8	4.7e3	3.3e3	6.3e3	1.4e3	0	*25e+0*	*15e+0*	*12e+1*	1.3e3
1	0	*76e−1*	*42e−1*	*46e+0*	6.3e2	.	.	.	.	.
1e−1	.	.	.	.	.	.	.	.	.	.
1e−3	.	.	.	.	.	.	.	.	.	.
1e−5	.	.	.	.	.	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.

	f123 in 5-D, N=15, mFE=5032					**f123 in 20-D**, N=15, mFE=756				
Δf	#	ERT	10%	90%	RT_{succ}	#	ERT	10%	90%	RT_{succ}
10	5	3.3e3	1.8e3	5.1e3	1.5e3	0	*27e+0*	*14e+0*	*94e+0*	1.8e2
1	0	*13e+0*	*72e−1*	*24e+0*	5.0e1	.	.	.	.	.
1e−1	.	.	.	.	.	.	.	.	.	.
1e−3	.	.	.	.	.	.	.	.	.	.
1e−5	.	.	.	.	.	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.

	f124 in 5-D, N=15, mFE=5007					**f124 in 20-D**, N=15, mFE=20008				
Δf	#	ERT	10%	90%	RT_{succ}	#	ERT	10%	90%	RT_{succ}
10	13	1.4e3	7.2e2	2.0e3	9.6e2	5	4.6e4	3.9e4	5.2e4	1.3e4
1	2	3.2e4	2.8e4	3.6e4	5.0e3	0	*11e+0*	*74e−1*	*44e+0*	3.5e3
1e−1	0	*43e−1*	*80e−2*	*10e+0*	1.1e3	.	.	.	.	.
1e−3	.	.	.	.	.	.	.	.	.	.
1e−5	.	.	.	.	.	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.

	f125 in 5-D, N=15, mFE=5007					**f125 in 20-D**, N=15, mFE=20007				
Δf	#	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	2.1e2	4.4e1	3.8e2	2.1e2	9	1.5e4	9.9e3	2.1e4	1.1e4
1e−1	4	1.4e4	1.2e4	1.7e4	2.7e3	0	*95e−2*	*46e−2*	*19e−1*	5.6e3
1e−3	0	*18e−2*	*33e−3*	*30e−2*	8.9e2	.	.	.	.	.
1e−5	.	.	.	.	.	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.

	f126 in 5-D, N=15, mFE=5004					**f126 in 20-D**, N=15, mFE=20008				
Δf	#	ERT	10%	90%	RT_{succ}	#	ERT	10%	90%	RT_{succ}
10	14	3.6e2	1.1e0	7.2e2	3.6e2	15	1.0e0	1.0e0	1.0e0	1.0e0
1	7	6.4e3	4.8e3	8.1e3	2.1e3	0	*31e−1*	*17e−1*	*49e−1*	5.0e3
1e−1	0	*12e−1*	*32e−2*	*68e−1*	1.3e3	.	.	.	.	.
1e−3	.	.	.	.	.	.	.	.	.	.
1e−5	.	.	.	.	.	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.

	f127 in 5-D, N=15, mFE=5004					**f127 in 20-D**, N=15, mFE=20003				
Δf	#	ERT	10%	90%	RT_{succ}	#	ERT	10%	90%	RT_{succ}
10	15	5.5e0	1.0e0	1.0e1	5.5e0	15	1.0e0	1.0e0	1.0e0	1.0e0
1	12	1.5e3	7.0e2	2.3e3	1.5e3	1	2.8e5	2.6e5	3.0e5	2.0e4
1e−1	0	*38e−2*	*21e−2*	*19e−1*	3.5e2	0	*19e−1*	*12e−1*	*24e−1*	1.0e4
1e−3	.	.	.	.	.	.	.	.	.	.
1e−5	.	.	.	.	.	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.

	f128 in 5-D, N=15, mFE=5007					**f128 in 20-D**, N=15, mFE=20002				
Δf	#	ERT	10%	90%	RT_{succ}	#	ERT	10%	90%	RT_{succ}
10	0	*34e+0*	*17e+0*	*64e+0*	5.0e1	0	*76e+0*	*69e+0*	*83e+0*	5.0e2
1	.	.	.	.	.	.	.	.	.	.
1e−1	.	.	.	.	.	.	.	.	.	.
1e−3	.	.	.	.	.	.	.	.	.	.
1e−5	.	.	.	.	.	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.

	f129 in 5-D, N=15, mFE=5003					**f129 in 20-D**, N=15, mFE=20005				
Δf	#	ERT	10%	90%	RT_{succ}	#	ERT	10%	90%	RT_{succ}
10	0	*20e+0*	*13e+0*	*65e+0*	1.3e3	0	*76e+0*	*73e+0*	*85e+0*	1.0e0
1	.	.	.	.	.	.	.	.	.	.
1e−1	.	.	.	.	.	.	.	.	.	.
1e−3	.	.	.	.	.	.	.	.	.	.
1e−5	.	.	.	.	.	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.

	f130 in 5-D, N=15, mFE=5003					**f130 in 20-D**, N=15, mFE=20003				
Δf	#	ERT	10%	90%	RT_{succ}	#	ERT	10%	90%	RT_{succ}
10	6	8.1e3	6.4e3	1.0e4	3.3e3	0	*76e+0*	*72e+0*	*81e+0*	6.3e3
1	1	7.0e4	6.5e4	7.5e4	5.0e3	.	.	.	.	.
1e−1	1	7.0e4	6.6e4	7.5e4	5.0e3	.	.	.	.	.
1e−3	0	*16e+0*	*11e−1*	*36e+0*	1.6e3	.	.	.	.	.
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 $f_{\rm opt} + \Delta f$ (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 $f_{\rm opt} + \Delta f$ 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.

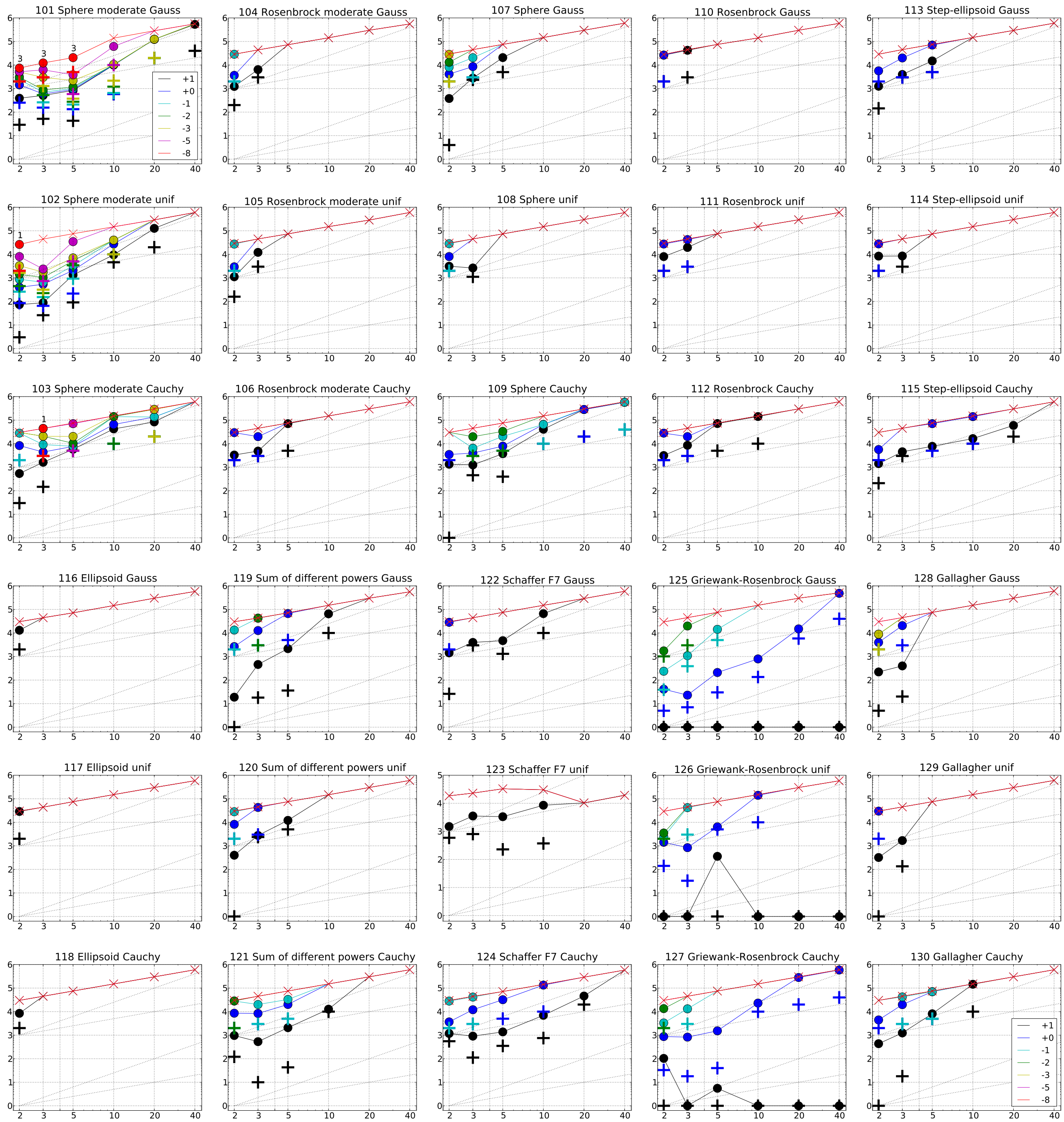

Figure 4: Expected Running Time (ERT, ●) to reach $f_{\mathrm{opt}} + \Delta f$ and median number of function evaluations of successful trials (+), shown for $\Delta f = 10, 1, 10^{-1}, 10^{-2}, 10^{-3}, 10^{-5}, 10^{-8}$ (the exponent is given in the legend of f_{101} and f_{130}) versus dimension in 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.

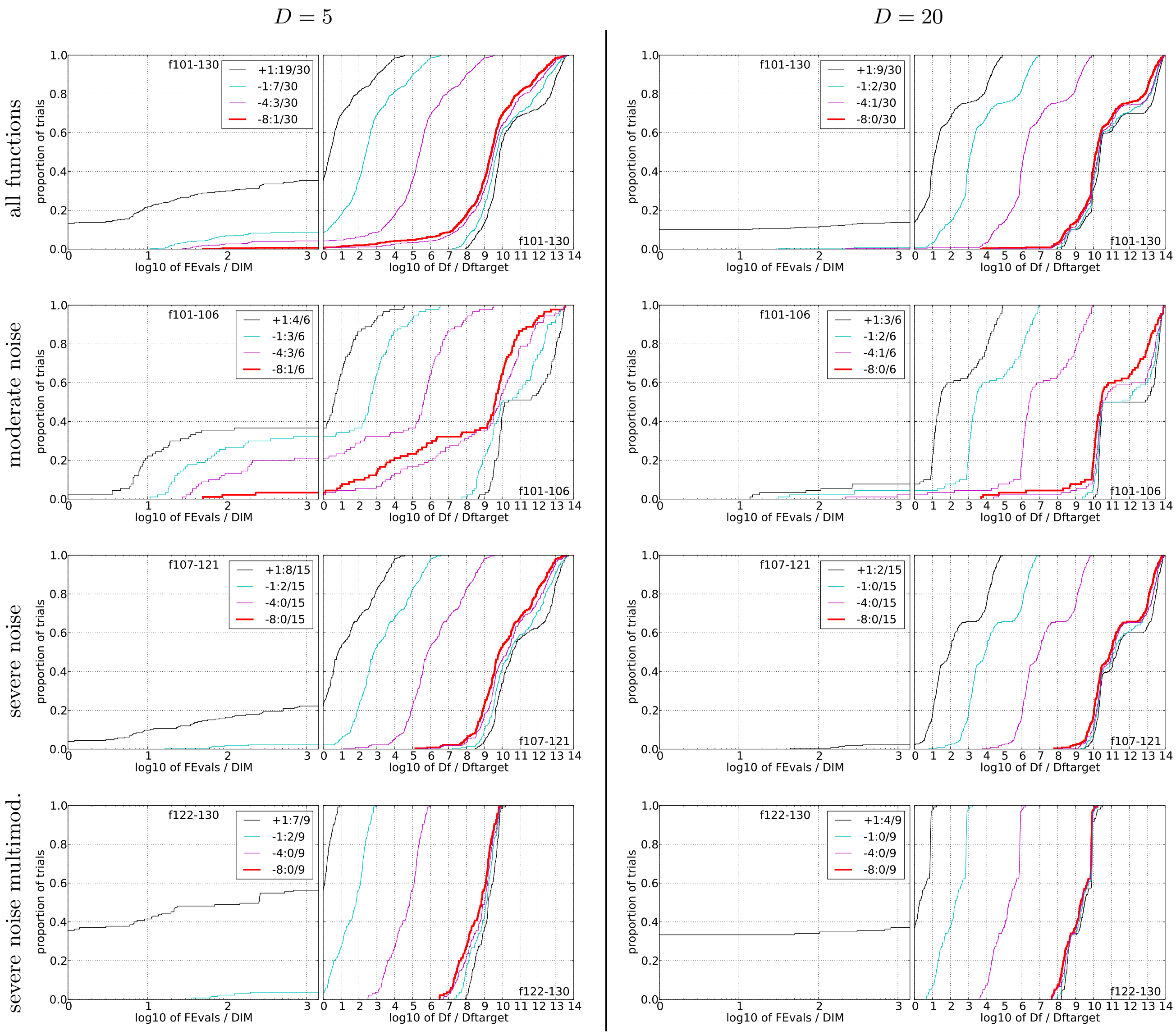

Figure 5: Empirical cumulative distribution functions (ECDFs), plotting the fraction of trials versus running time (left subplots) or versus Δf (right subplots). The thick red line represents the best achieved results. Left subplots: ECDF of the running time (number of function evaluations), divided by search space dimension D, to fall below $f_{\rm opt} + \Delta f$ 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.