A robust simplex algorithm for online optimization

Xiaobiao Huang*
SLAC National Accelerator Laboratory, Menlo Park, CA 94025, USA
(Dated: June 12, 2018)

A new optimization algorithm is introduced for online optimization applications. The algorithm was modified from the popular Nelder-Mead simplex method to make it noise aware and noise resistant. Simulation with an analytic function is used to demonstrate its performance. The algorithm has been successfully tested in experiments, which showed that the algorithm is robust for optimization problems with complex functional dependence, high cross-coupling between parameters, and high noise. Advantages of the new algorithm include high efficiency and that it does not require prior knowledge of the parameter space such as an initial conjugate direction set.

PACS numbers: 29.50.+v, 29.85.-c

I. INTRODUCTION

Large, complex machines such as accelerators are usually built according to exquisitely studied and optimized designs and are expected to perform in a way as predicted in the design studies. However, in reality there are always random and systematic errors in an actual machine that are not included in the corresponding design model and these errors will cause deviations of the behaviors of the machine from the model. Compensation of these errors is necessary for the machine to attain the optimal performance.

Traditionally the desired error compensation approach is to use diagnostics to monitor or probe the beam and to use data acquired by these diagnostics to determine the actual errors or a set of changes to actuators (i.e., knobs) for corrections. This approach, however, may not always be possible as there may be a lack of diagnostics, an undetermined correction target, or a lack of causal relationship between the diagnostic measurements and the performance measures. In such cases, tuning knobs directly to improve machine performance is inevitable.

Manual tuning has been common since the early days of accelerators. In the era of computerized controls, automated tuning has become possible. Traditional algorithms, such as the Nelder-Mead simplex method [1], have been implemented in accelerator controls [2]. More recently, an exploration of suitable online optimization algorithms has led to the invention of the robust conjugate direction search method (RCDS) [3], which has been proven to be an effective method for automated accelerator tuning through successful applications at many laboratories [4–10].

The RCDS algorithm is an efficient parameter scan method for multi-variable optimization problems without significant cross-coupling between the decision variables. For problems with severe cross-coupling, the algorithm, in theory, can build up a conjugate direction set in the parameter space to gain high efficiency [11]. However, in realistic online applications, the algorithm usually does not run enough iterations to substantially benefit from the scheme. Therefore, supplying an initial conjugate direction set (which may be obtained with a model) to the algorithm is very important. For example, Ref. [3] showed that, for the storage ring coupling minimization problem, RCDS efficiency is much higher with the initial conjugate direction set calculated with the lattice model than by starting with directions along individual decision variables (skew quadrupoles).

On the other hand, Ref. [3] showed that the Nelder-Mead simplex method is very efficient without prior knowledge of the parameter space. It was also shown that the deficiency of the simplex method for online optimization is its susceptibility to noise in function evaluation. As soon as the noise starts to alter the results of function value comparisons between the simplex vertices, the algorithm breaks down and fails to converge to the optimum. The above observations have prompted us to modify the simplex method to improve its robustness against noise.

In this study we propose a robust simplex algorithm that is suitable for the optimization of noisy functions. This algorithm takes the noise level into consideration in function value comparisons, takes additional measurements if necessary to reduce noise, uses local curve fitting when direct comparisons do not yield definitive operation decisions, and explores multiple directions in an iteration. These modifications make the algorithm significantly more robust against noise. We have tested the new algorithm in simulation with the analytic Rosenbrock function [12]. We have also tested it in experiments on the SPEAR3 storage ring for the coupling minimization and the kicker bump matching problems. The results demonstrated that the new method is robust against noise and is efficient, even for problems with high cross-coupling between parameters.

In Section II we describe the robust simplex algorithm. Section III discusses the simulation test of the algorithm. Experimental results for both the coupling correction problem and the kicker bump matching problem are presented in Section IV. The conclusion is given in Section V.

* xiahuang@slac.stanford.edu

II. THE ROBUST SIMPLEX ALGORITHM

In this paper we deal with multi-variable, single objective optimization problems. For definitiveness, we assume minimization problems. In online machine tuning applications, the decision variables naturally have limited valid ranges. As in Ref. [3], we normalize each variable to the range $[0, 1]$. Each point in the n-dimensional parameter space is represented by a vector $\mathbf{X} = (x_1, x_2, \cdots, x_n)$, where n is the number of decision variables. The goal of the algorithm is to find a point, $\mathbf{X}_{\min}$, where the objective function value $f(\mathbf{X}_{\min})$ is lower than the value at any other point.

A. The original simplex algorithm

The Nelder-Mead simplex algorithm does not constrain the ranges of the variables. A simple modification to implement variable range constraint is to set any component of the solution vector $\mathbf{X}$ to the nearest range limit, which is 0 or 1.

The simplex algorithm follows a simple and elegant paradigm. It operates with a simplex in the parameter space, which consists of $n+1$ vertices. The initial simplex may be rebuilt by taking a small step along each axis from the starting point to make n new points.

After the objective function values for each vertex is evaluated, the original simplex algorithm executes the following steps iteratively [1]:

1: Sort the function values for the $n + 1$ vertices, with $f_1 < \cdots < f_n < f_{n+1}$. The corresponding vertices are located at $\mathbf{X}_1$, $\cdots$, $\mathbf{X}_n$, $\mathbf{X}_{n+1}$.

2: Define the center point on the simplex face that is opposite to vertex $\mathbf{X}_{n+1}$, $\mathbf{X}_c = \frac{1}{n}\sum_{i=1}^{n} \mathbf{X}_i$. The algorithm may evaluate the function values at several points on the line connecting the points $\mathbf{X}_{n+1}$ and $\mathbf{X}_c$. These points are defined as follows:

reflection: $\mathbf{X}_r = \mathbf{X}_c + (\mathbf{X}_c - \mathbf{X}_{n+1})$.

expansion: $\mathbf{X}_e = \mathbf{X}_c + 2(\mathbf{X}_c - \mathbf{X}_{n+1})$.

inner contraction: $\mathbf{X}_{\rm ic} = \mathbf{X}_c - \frac{1}{2}(\mathbf{X}_c - \mathbf{X}_{n+1})$.

outer contraction: $\mathbf{X}_{\rm oc} = \mathbf{X}_c + \frac{1}{2}(\mathbf{X}_c - \mathbf{X}_{n+1})$.

First evaluate the function value at the reflection point, $f_r = f(\mathbf{X}_r)$. If $f_1 < f_r < f_n$, replace vertex $\mathbf{X}_{n+1}$ with $\mathbf{X}_r$ and terminate the iteration.

3: If $f_r < f_1$, evaluate the function value at the expansion point, $f_e = f(\mathbf{X}_e)$, replace vertex $\mathbf{X}_{n+1}$ with $\mathbf{X}_r$ or $\mathbf{X}_e$, whichever has the smaller function value, and terminate the iteration.

4: If $f_n < f_r < f_{n+1}$, evaluate $f_{\rm oc} = f(\mathbf{X}_{\rm oc})$. If $f_{\rm oc} < f_r$, replace vertex $\mathbf{X}_{n+1}$ with $\mathbf{X}_{\rm oc}$ and terminate the iteration.

5: If $f_r > f_{n+1}$, evaluate $f_{\rm ic} = f(\mathbf{X}_{\rm ic})$. If $f_{\rm ic} < f_{n+1}$, replace vertex $\mathbf{X}_{n+1}$ with $\mathbf{X}_{\rm ic}$ and terminate the iteration.

6: When none of the previous steps terminate the iteration, perform a shrink toward $\mathbf{X}_1$, the vertex with the minimum function value, i.e., replace vertex $\mathbf{X}_2$ through $\mathbf{X}_{n+1}$ with new points

$$\mathbf{X}'_i = \mathbf{X}_1 + \frac{1}{2}(\mathbf{X}_i - \mathbf{X}_1),$$

where $i = 2, 3, \cdots, n + 1$. Terminate the iteration.

After each iteration, the algorithm goes back to step 1 for the next iteration. The algorithm may terminate after a pre-specified number of function evaluations or after the difference between the minimum and the maximum of the vertex function values is below a target value.

B. The robust simplex method

The original simplex method uses function value comparisons at almost every step. The outcomes of these comparisons provide information about the objective function in the vicinity of the simplex which aids the algorithm in choosing the search path. When noise contaminates the function values, the comparison outcomes may be changed, which will distort the function information, lead to a wrong search direction, and prevent the algorithm from converging to the minimum. On the other hand, the algorithm has high efficiency in searching for the minimum when there is no noise. Therefore, the appropriate approach of modifying the algorithm to improve its robustness against noise would be to preserve the actions wherever noise does not modify the comparison results and to reduce noise or to change the behavior when the comparison results become corrupted due to noise.

During an iteration, the main actions of the simplex method are to sample the objective function along the line connecting $\mathbf{X}_{n+1}$ and $\mathbf{X}_c$ at the specified positions. If the reflection and contraction operations result in a new point with function value lower than the current maximum vertex value, the new point will replace the current maximum vertex. Noise in function values may introduce two problems here. First, it may affect the sorting results of the vertex function values. The differences between the largest function values of the vertices may be below the noise level and thus there is no unambiguous maximum value vertex. Second, the sequence of actions in reflection, expansion, and contraction may be changed by noise.

A straightforward remedy for the ambiguity in comparison results would be to increase the number of samples at the points involved. However, we do not want to increase the sample numbers at all points since in many

cases the comparison results are not ambiguous. A sensible approach would be to increase the number of samples as needed using the noise level and the differences between the two values in a comparison as the guide. It is noted that the average value of N samples of a normal distribution $\mathcal{N}(\mu, \sigma^2)$, with expectation μ and standard deviation σ, obeys the normal distribution $\mathcal{N}(\mu, \sigma^2/N)$. If we draw N_1 and N_2 samples from two normal distributions, $\mathcal{N}(\mu_1, \sigma_1^2)$ and $\mathcal{N}(\mu_2, \sigma_2^2)$, and calculate their average values, $\bar{X}_1$ and $\bar{X}_2$, respectively, the difference between the two average values, $\bar{X}_1 - \bar{X}_2$, obeys the normal distribution

$$\mathcal{N}(\mu_1 - \mu_2, \Sigma^2), \quad \text{with } \Sigma^2 = \frac{\sigma_1^2}{N_1} + \frac{\sigma_2^2}{N_2}$$

If the absolute value of $\mu_1 - \mu_2$ is substantially larger than Σ, the sign of $\bar{X}_1 - \bar{X}_2$ is a good estimate of the sign of $\mu_1 - \mu_2$. For example, if $|\mu_1 - \mu_2| = \Sigma$, the chance of $\mu_1 - \mu_2$ and $\bar{X}_1 - \bar{X}_2$ having the same sign is 84%; if $|\mu_1 - \mu_2| = 1.4\Sigma$, the chance is 92%.

When applying the above statistical theory to the comparison of function values at two vertices, it is assumed that the standard deviation at every point in the parameter space is equal; and, we will use $\bar{X}_1 - \bar{X}_2$ as an estimate of $\mu_1 - \mu_2$. Therefore, when necessary we will increase the number of samples, N_1 and N_2, until

$$|\bar{X}_1 - \bar{X}_2| \geq M_1 \sigma \sqrt{\frac{1}{N_1} + \frac{1}{N_2}}, \tag{1}$$

or an upper limit, $N_{\max}$, for the total number of evaluations per point is reached, where M_1 is a numerical value which we choose to be 1.4. Upper limits for N_1 and N_2 are set to avoid excessive function evaluations. The actual value of the upper limit may depend on the noise level and the nature of the function terrain. High noise levels and complicated terrains would require more averaging.

If a comparison result of two function values is obtained with condition (1) satisfied, we call the result definitive, otherwise ambiguous. In cases the determination of the maximum-value vertex is ambiguous, instead of trying to pick out the actual maximum-value vertex, the algorithm collects a set of vertices with the largest values and performs the reflection, expansion, contraction operations on each of them until a significant reduction (i.e., above the noise level) of the maximum vertex function value is achieved.

In the reflection, expansion, and contraction operations, the corresponding vertex is replaced by a new point only if the comparisons leading to it are definitive. If no vertex replacement takes place with the operations along the line of the vertex and the center point on its opposite face, a quadratic fit of five points on the line is performed to improve the accuracy of determining the next step.

Combining the modifications, we devised a robust simplex (RSimplex) algorithm that is suitable for online optimization. The algorithm requires the rms noise level of the objective function, σ, as an input parameter. It also takes an iterative approach to change the simplex using function values at the vertices as a guide. When making a comparison of function values at two points, unless noted otherwise, the procedure described in the above to reach a definitive result is applied, subject to the upper limit of the number of evaluations for each point. Details of the steps in one iteration are described in the following.

1: Sort the vertex function values in the ascending order and identify the maximum value vertex group, $G_{\max}$, with

$$G_{\max} = \{\mathbf{X}_{n+1}, \mathbf{X}_n, \cdots, \mathbf{X}_{n-m_1+2}\},$$

where vertices $\mathbf{X}_n$ through $\mathbf{X}_{n-m_1+2}$ have function values that are too close to $f(\mathbf{X}_{n+1})$ to be definitively ruled out as the maximum value vertex. Upper limits may be given to the size of the group, m_1. For example, m_1 may be no more than 4.

2: Perform reflection/expansion and contraction operations for members vertices of $G_{\max}$ sequentially, terminate the iteration if a vertex replacement takes place.

The reflection/expansion and contraction operations are described in more details below. For each member vertex of $G_{\max}$, say $\mathbf{X}_i$, calculate the function value at its reflection point, f_r. If $f_r < f_1$, also evaluate at the expansion point and compare the functions values at the reflection point and the expansion point, use the point with the lower function value to replace $\mathbf{X}_i$ and terminate the iteration if the comparison is definitive; otherwise, evaluate the middle point between the two points, use it to replace $\mathbf{X}_i$, and terminate the iteration.

Otherwise, compare f_r and $f_n \equiv f(\mathbf{X}_n)$. If f_r is definitively lower than f_n, use $\mathbf{X}_r$ to replace $\mathbf{X}_i$; otherwise, move on to the contraction operation as described in the next paragraph.

In the contraction operation, if $f_r > f_i$ definitively, evaluate the inside contraction point; if, instead, $f_r < f_{n+1}$ definitively, evaluate the outside contraction point; otherwise evaluate both the inside and outside contraction points and the center of mass point $\mathbf{X}_c$ and perform a quadratic fit for a more accurate examination. If either the inside contraction yields $f_{\text{ic}} < f_{n+1}$ or the outside contraction yields $f_{\text{oc}} < f_{n+1}$ definitively, use the corresponding contraction point to replace vertex $\mathbf{X}_i$. Otherwise perform the quadratic fit.

In the quadratic fit, function values at five points, $\mathbf{X}_i$ (i.e., the member vertex in $G_{\max}$) and its corresponding $\mathbf{X}_{\text{ic}}$, $\mathbf{X}_c$, $\mathbf{X}_{\text{oc}}$, and $\mathbf{X}_r$ are fitted to a function

$$y = a\alpha^2 + b\alpha + c,$$

where $\alpha = -1$, -0.5, 0, 0.5, 1 for the five points, respectively, and y is the function value. From the

fitting we also get the uncertainty of the fitting parameters. If each of the five points is only evaluated once, we have $\sigma_a = 1.07\sigma$, $\sigma_b = 0.63\sigma$, and $\sigma_c = 0.70\sigma$. The fitted function value difference between points $\mathbf{X}_i$ and $\mathbf{X}_{\rm ic}$ is $f_i - f_{\rm ic} = \frac{3}{4}a - \frac{1}{2}b$. If this value is larger than $M_1\sigma_c$ and $b > 0$, we use the inside contraction point to replace $\mathbf{X}_i$. If $b < 0$ and $\frac{3}{4}a + \frac{1}{2}b > M_1\sigma_c$, we use the outside contraction point to replace $\mathbf{X}_i$. Terminate the iteration if vertex $\mathbf{X}_i$ is replaced.

3: If no vertex replacement takes place in step 2, and if the difference between the maximum and minimum vertex function values, $f_{n+1} - f_1$, is above $M_2\sigma$, perform a shrink toward the vertex with the minimum function value. Here M_2 is a numerical value which may be chosen to be 2.0. The M_2 requirement is imposed to avoid the reduction of the simplex size to a level when comparison operations are swamped by noise. Note there could be ambiguity as to which vertex has the minimum value. Comparisons between a few leading candidates for a definitive choice are performed, subject to the upper limit of the number of function evaluations for each vertex. After the shrink operation, move to step 1 for the next iteration.

The values $M_1 = 1.4$ or $M_2 = 2.0$ are empirically chosen and are somewhat arbitrary. The optimal values may be obtained from a statistical analysis.

The size of the simplex decreases every time a contraction or shrink operation is performed. During optimization the simplex may shrink in size to a point such that the differences between the vertex function values are not significantly higher than the noise level. At this point, if no gain is being made in reducing either f_{n+1} or f_1, one may re-build the simplex by using the current minimum as the starting point. The vertex with the current minimum is kept in the simplex and the other vertices are replaced with points shifted in one axis from the minimum in the same fashion as is done for the construction of the initial simplex. The step size of the parameter shift may be equal to or a fraction of the initial step size. This could allow the search algorithm to jump out a local minimum, although there is no guarantee that the algorithm can find the global minimum.

Additional exploration of the parameter space may also be introduced when the simplex has become too small compared to the noise level. Such exploration can be a search over a direction that is perpendicular to a simplex face with the robust 1-dimensional optimizer as found in Ref. [3], or some sorts of stochastic exploration around the minimum value vertices.

It is worth pointing out that both the original Nelder-Mead simplex method and the RSimplex method are single objective algorithms and tend to converge to nearby local extrema. Multi-objective genetic algorithms (MOGA) [13, 14] or multi-objective particle swarm optimization (MOPSO) [4, 15] algorithms could be used when a multi-objective application or a global search over a parameter space with many local extrema are desired, although caution should be given to the fact that the performance of MOGA can be affected by function evaluation noise [3].

III. SIMULATION

To test the performance of the modified simplex program, we did a simulation study using the analytic Rosenbrock function [12]. The Rosenbrock function with n variables is defined to be

$$f = \sum_{i=1}^{n-1} 100(x_i - x_{i-1}^2)^2 + (1 - x_i)^2. \qquad (2)$$

In the tests we set $n = 6$ and the parameter range of all 6 variables to be within $[-5, 5]$. The global minimum is $f = 0$, which is achieved when $x_i = 1$ for $i = 1, 2, \cdots, 6$.

The initial solution is chosen to be the origin, with $x_i = 0$ for all 6 variables and a corresponding function value of $f = 5$. Without noise the original simplex algorithm (Nelder-Mead) converges to the minimum with about 600 function evaluations. The step length for the initial simplex is 2. When random Gaussian noise with a standard deviation $\sigma = 0.01$ is added to the function, the Nelder-Mead algorithm typically does not converge to the minimum. Instead, it converges to solutions with function values between 0 and 4.5.

When the modified simplex algorithm is applied, with $M_1 = 1.4$ and $M_2 = 2.0$, and an upper limit of function evaluations per vertex $N_{\rm max} = 3$, the minimum function values achieved are significantly closer to the minimum. FIG. 1 shows the minimum function values obtained within 1000 function evaluations for 100 repeated runs, sorted in the ascending order, for the original simplex algorithm, the robust simplex without simplex rebuilding, and the robust simplex with simplex rebuilding. Simplex rebuilding is performed around the minimum function value vertex when $f_{n+1} - f_1 < M_2\sigma$ and the reductions of both the maximum and the minimum values in the last N iterations are less than 0.2σ. The side length of the rebuilt simplex is one half of the initial simplex. Also shown in FIG. 1 is the result for the original simplex method but with sample averaging for noise reduction, for which the objective function is the average of 3 evaluations.

FIG. 2 shows the histories of evaluated solutions for a typical case for the Nelder-Mead algorithm and the two variations of the robust simplex algorithm with a final minimum value that corresponds to the median value of the 100 cases. While the modified simplex method is more robust against noise, it can still be trapped by a sub-optimal solution. Rebuilding the simplex helps the algorithm break out from such a situation.

FIG. 1 and 2 clearly illustrate the benefits of the modified simplex algorithm. By taking extra samples when

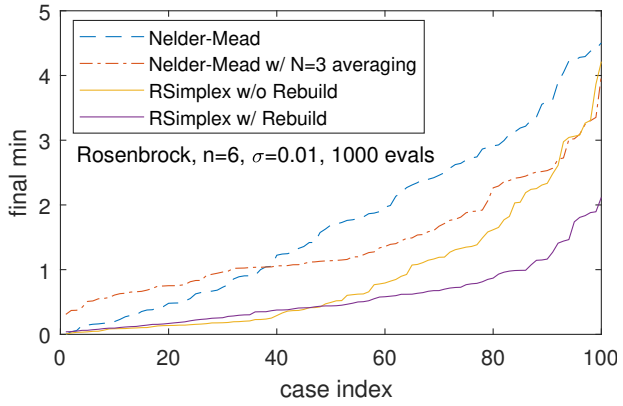

FIG. 1. Minimum function values in 1000 evaluations for the Rosenbrock problem in 100 optimization runs (sorted) with the Nelder-Mead simplex algorithm (blue dashed line), Nelder-Mead with $N = 3$ averaging (red dash-dot line), the robust simplex algorithm without simplex rebuilding (RSimplex w/o Rebuild, solid yellow line) and with simplex rebuilding (RSimplex w/ Rebuild, solid magenta line).

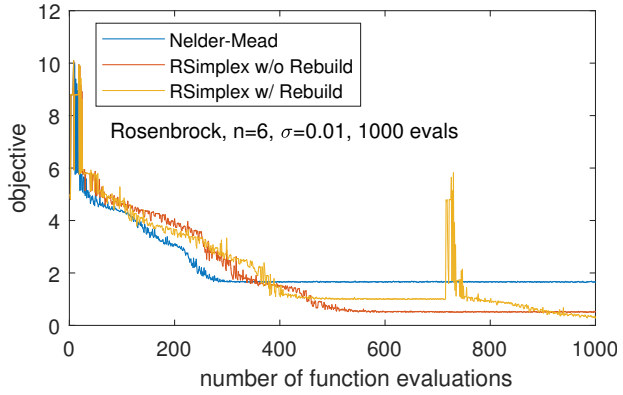

FIG. 2. History of the objective function values of all evaluated solutions for the Rosenbrock function optimization problem using three algorithms: Nelder-Mead simplex (blue line), RSimplex w/o simplex rebuilding (red line), and RSimplex w/ simplex rebuilding (yellow line).

needed and using local fitting to improve the accuracy of vertex comparisons, the modified algorithm is better able to find the optimum in a noisy environment.

IV. EXPERIMENTS

We have tested the robust simplex method on the SPEAR3 storage ring with two experiments. The first application is to minimize the vertical emittance with skew quadrupoles. The other is to minimize the transient oscillation on the stored beam by improving kicker bump matching. Both experiments were previously used to test the RCDS algorithm [3].

A. Coupling minimization

In an electron storage ring, the vertical emittance arises from various error sources, such as rolls of quadrupoles, vertical orbit distortion in sextupoles, and skew quadrupole components from insertion devices and other magnets. These errors can be compensated with skew quadrupole magnets. This is often referred to as coupling correction. In the SPEAR3 storage ring, we use 13 skew quadrupoles (which are actually windings on sextupole magnets) to correct coupling.

When the dominant beam loss is Touschek scattering loss, the beam loss rate is inversely proportional to the vertical beam size, which, in turn, is proportional to the square root of the vertical emittance. Skew quadrupoles typically do not affect beam lifetime in other ways. Therefore, maximizing beam loss rate with skew quadrupoles is equivalent to minimizing the vertical emittance.

In the experiment, beam loss over a 6-second period is converted to beam loss rate (in mA/min) to be used as the objective function (with a change of sign to make a minimization problem). The beam current is maintained at nearly 500 mA with top-off injection every 5 minutes. The initial setpoints of all 13 skew quadrupoles are set to zero. The corresponding loss rate is about 0.6 mA/min. The rms noise of the objective function is about 0.03 mA/min, which comes from the noise in the beam current measurement.

The ranges of skew quadrupole current set-points are from -20 to 20 Amp. The initial simplex is built by shifting from the initial point in the positive direction of each skew quadrupole by 10% of the whole range, or 4 Amp, to create the other 13 vertices. After the algorithm is launched, it moves the simplex in the parameter space toward the minimum without intervention.

The robust simplex algorithm converged to a minimum in about 200 function evaluations. The program was terminated after 260 evaluations were executed as no further improvement was made. The algorithm ran 91 iterations. The history of the normalized parameters in the optimization run is shown in FIG. 3, from which we can see the evaluation of the initial simplex vertices, the subsequent exploration of the parameter space, and finally the convergence toward the minimum.

The size of the simplex and the difference between the maximum and minimum values on the vertices varied during the iterations. FIG. 4 shows two dimension-less parameters defined as

$$u = \frac{f(\mathbf{X}_{n+1}) - f(\mathbf{X}_1)}{\sigma}, \qquad v = V^{\frac{1}{n}} \times 500, \qquad (3)$$

where V is the volume of the simplex, n is the dimension of the parameter space, and v is scaled arbitrarily for plotting. The u parameter serves as an indication of the simplex size relative to the function evaluation noise level. A small u (say, $u < 3.0$) means the function value comparison outcome would be frequently altered by ran-

dom noise. The v parameter represents the geometric dimension of the simplex. Changes in the v parameter indicate the nature of the operations being performed by the algorithm, as the volume of the simplex changes during expansion, contraction, or shrink operations. The first 44 iterations (about 100 evaluations) only applied reflection operations. There were two shrink operations toward the end.

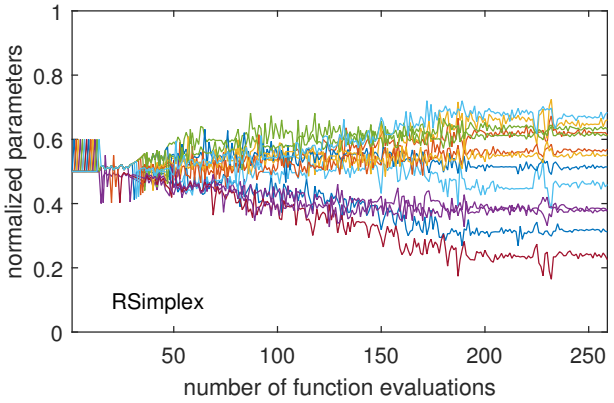

FIG. 3. Variation of the 13 skew quadrupole current setpoints (normalized to the range [0, 1]) for the coupling correction experiment using the RSimplex algorithm (w/o simplex rebuilding).

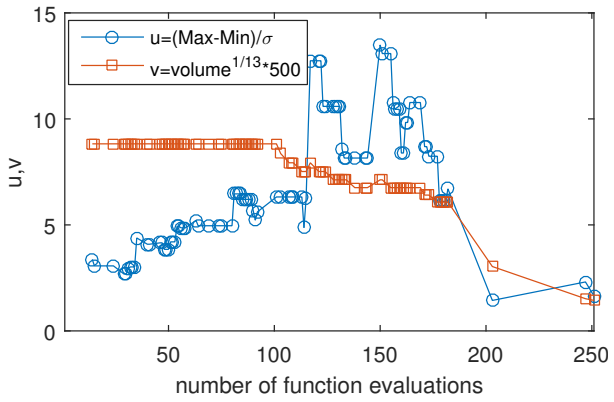

FIG. 4. Evolution of the simplex size as indicated by the u (circles) and v (squares) parameters defined in Eq. (3) during the RSimplex optimization run for the coupling correction experiment.

For comparison, we also tested the same optimization problem with the Nelder-Mead simplex method and the RCDS method. The history of the objective functions of all three algorithms are shown in FIG. 5. The Nelder-Mead simplex algorithm could not make any significant gains. As its simplex quickly shrank, it soon stopped making appreciable changes to the parameters. It was terminated after about 100 function evaluations as no gain was being made. The RCDS algorithm reached the same level of loss rate in about 120 evaluations. However, it benefited from the conjugate direction set that was calculated using the lattice model [3].

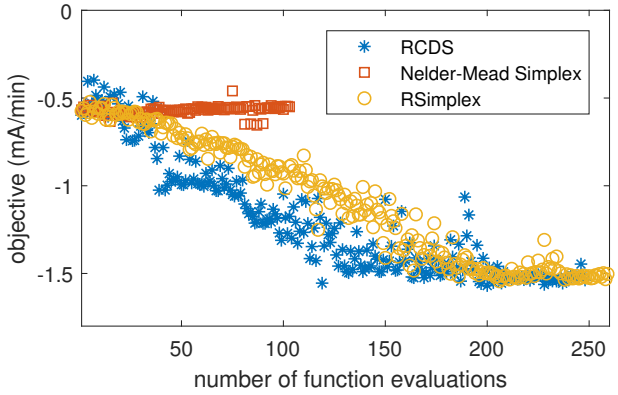

FIG. 5. History of the objective function (which is essentially the beam loss per minute with a negative sign) over the coupling minimization experiment for the three algorithms: RCDS, Nelder-Mead simplex, and RSimplex (w/o simplex rebuilding).

After the optimization, the skew quadrupoles were set to the best solutions found with RSimplex and RCDS, respectively. The corresponding loss rates were 1.55 and 1.56 mA/min, respectively. The skew quadrupole setting for coupling correction obtained with LOCO [16], the orbit response matrix based method, was also applied to the machine. The resulting loss rate was 1.41 mA/min, lower than the solutions found with the optimization algorithms.

B. Kicker bump matching

The robust simplex algorithm is also applied to optimize the kicker bump matching problem on SPEAR3. This problem was previously used to test the RCDS algorithm [3] and the extremum seeking (ES) algorithm [17]. The goal is to minimize the transient oscillation of the stored beam after the three injection kickers are fired. The kick amplitude, pulse width, and pulse delay for each kicker can be changed. The parameters for one of the kickers, K3, are held constant, while the parameters for the other two kickers are used as optimization knobs. There are two skew quadrupoles between the kickers, which affect the horizontal to vertical coupling. These two skew quadrupoles are also used as optimization knobs to help reduce vertical oscillation. There are a total of 8 knobs.

The objective function is $\sigma_x + 3\sigma_y$, where $\sigma_{x,y}$ are the rms of the horizontal and vertical turn-by-turn orbit readings on a beam position monitor (BPM) for the first 256 turns after the kickers are fired. A weight factor of 3 is given to the vertical plane because user experiments are more sensitive to vertical oscillations. The initial oscillation amplitudes correspond to approximately

$\sigma_x = 100$ μm and $\sigma_y = 35$ μm. The noise sigma for the objective function is $\sigma = 3$ μm.

Four algorithms, the robust simplex, the Nelder-Mead simplex, RCDS, and the ES were applied. The Nelder-Mead simplex method also worked in this experiment because the cross-coupling between the decision variables is not severe and the function terrain in the parameter space is relatively simple. FIG. 6 shows the history of the u, v parameters defined similarly as in the coupling minimization problem. It is noted that the u parameter remains at large values ($u > 10.0$) before the algorithm converged, which could explain why here the RSimplex method behaves similarly to the Nelder-Mead simplex method. The difference of function values on the simplex vertices is significantly higher than the noise level before it converged. The modifications we introduced in the robust simplex do not need to kick in if there is no ambiguity in function value comparisons.

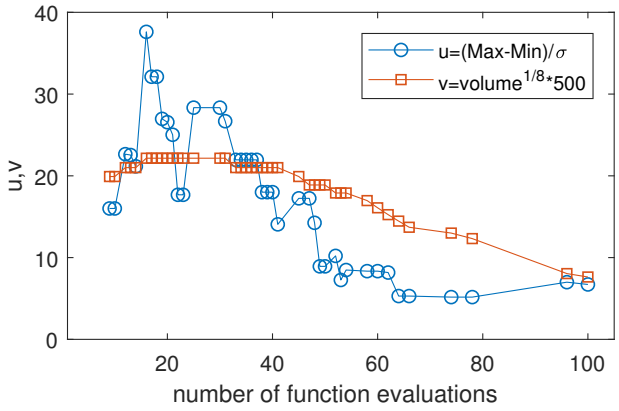

FIG. 6. Evolution of the simplex size as indicated by the u (circles) and v (squares) parameters defined in Eq. (3) during the RSimplex optimization run for the kicker bump matching experiment.

The history of the objective function values during the experiments for all algorithms are shown in FIG. 7. The robust simplex, the original simplex, and RCDS reached the same minimum level of objective function with about the same number of function evaluations. This demonstrates the fact that the robust simplex does not make unnecessary additional steps in the converging process. No initial conjugate direction set was supplied to the RCDS algorithm in this experiment. The results of the ES algorithm on the same problem, using the control parameter values in Ref. [17], are presented for comparison.

V. CONCLUSION

We modified the original Nelder-Mead simplex algorithm for online optimization. The new algorithm (robust simplex, or RSimplex) takes extra samples for noise reduction when statistically the comparisons of function values do not yield definitive results, and makes additional changes to the operations to improve the accuracy of decision making and to further explore the parameter space. The new algorithm is significantly more robust against noise than the original simplex algorithm in the optimization of complex functions. Different from the RCDS algorithm, which is also robust against noise, the robust simplex algorithm does not need any prior information about the objective function in order to be efficient for problems with high cross-coupling between decision variables.

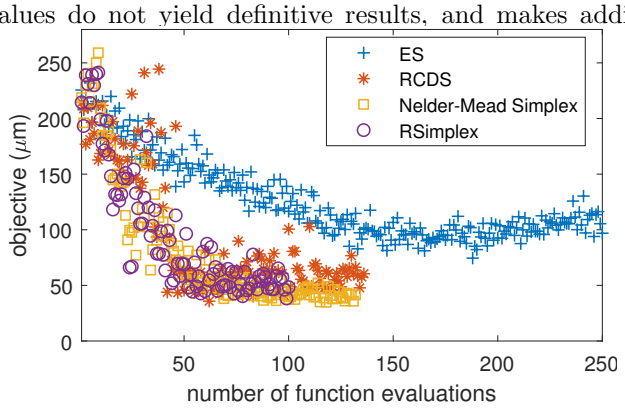

FIG. 7. History of the objective function ($\sigma_x + 3\sigma_y$ of the turn-by-turn beam position oscillation in the first 256 turns after the kickers are fired) for the kicker bump matching experiments for four optimization algorithms: Extremum Seeking (ES, '+'), RCDS ('*'), Nelder-Mead simplex (squares), and RSimplex (w/o simplex rebuilding, circles).

The new algorithm has been tested with simulations using an analytic function and demonstrated with experiments on the SPEAR3 storage ring. The coupling minimization experiment showed that the robust simplex algorithm can find the optimum setting despite significant cross-coupling between the decision variables, complex function terrain, and high noise levels. In the less challenging problem of kicker bump matching, both the robust simplex method and the original method worked with the same efficiency.

ACKNOWLEDGMENTS

Work was supported by the U.S. Department of Energy, Office of Science, Office of Basic Energy Sciences, under Contract No. DE-AC02-76SF00515.

[1] J. A. Nelder and R. Mead, The Computer Journal **7**, 308 (1965).

[2] L. Emery, M. Borland, and H. Shang, in *Proceedings of PAC03* (Portland, Oregon, USA, 2003) pp. 2330–2332.

[3] X. Huang, J. Corbett, J. Safranek, and J. Wu, Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment **726**, 77 (2013).

[4] X. Huang and J. Safranek, Phys. Rev. ST Accel. Beams **18**, 084001 (2015).

[5] H.-F. Ji, Y. Jiao, S. Wang, D.-H. Ji, C.-H. Yu, Y. Zhang, and X. Huang, Chinese Physics C **39**, 127006 (2015).

[6] S. Liuzzo, N. Carmignani, L. Farvacque, P. T. Nash, B., P. Raimondi, R. Versteegen, and S. M. White, in *Proceedings of IPAC2016* (Busan, Korea, 2016) pp. 3420–3422.

[7] I. Martin, M. Apollonio, and R. Bartolini, in *Proceedings of IPAC2016* (Busan, Korea, 2016) pp. 3381–3383.

[8] G. Wang, W. Cheng, X. Yang, J. Choi, and T. Shaftan, in *Proceedings of IPAC2017* (Copenhagen, Denmark, 2017) pp. 4683–4685.

[9] T. Pulampong, P. Klysubun, S. Kongtawong, S. Krainara, and S. Sudmuang, in *Proceedings of IPAC2017* (Copenhagen, Denmark, 2017) pp. 4086–4088.

[10] W. F. Bergan, A. C. Bartnik, I. V. Bazarov, H. He, D. L. Rubin, and J. P. Sethna, in *Proceedings of IPAC2017* (Copenhagen, Denmark, 2017) pp. 2418–2420.

[11] M. J. D. Powell, The Computer Journal **7**, 155 (1964).

[12] H. Rosenbrock, The Computer Journal **3**, 175184 (1960).

[13] I. V. Bazarov and C. K. Sinclair, Phys. Rev. ST Accel. Beams **8**, 034202 (2005).

[14] K. Tian, J. Safranek, and Y. Yan, Phys. Rev. ST Accel. Beams **17**, 020703 (2014).

[15] X. Pang and L. Rybarcyk, Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment **741**, 124 (2014).

[16] J. Safranek, Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment **388**, 27 (1997).

[17] A. Sheinker, X. Huang, and J. Wu, IEEE Transctions on control systems technology **26**, 336 (2018).