

AN OPTIMIZED STEP-SIZE RANDOM SEARCH (OSSRS)

B.V. SHEELA *

Aerodynamics Division, National Aeronautical Laboratory, Bangalore 560 017, India

Received 3 December 1976

A random search method, Optimized Step-Size Random Search (OSSRS), for function minimization is proposed. In this method the step-size is determined by fitting a quadratic for the function value in terms of the step-size in each of the random directions. The random direction is generated using a pseudorandom number generator with a normal distribution of zero mean and a given standard deviation.

1. Introduction

The general unconstrained optimization problem can be stated as:

$$\text{minimize } f(\mathbf{x}), \quad \mathbf{x} = \{x_1, x_2, \dots, x_n\} \in \mathbb{R}^n,$$

where $f(\mathbf{x})$ is a nonlinear function of the variables x_1, x_2, \dots, x_n . Several methods have been proposed in the literature to solve these unconstrained optimization problems, which come under the “direct search” category. The general philosophy underlying any direct search method is to successively obtain better points in the search space by generating new directions and determining step-sizes to be taken in these directions. Random search methods are that class of techniques wherein the directions are generated by pseudorandom number generators. Many methods have been proposed towards this end. These include the Randomized Pattern Search (RPS) of Lawrence and Steiglitz [1] and the Adaptive Random Search Algorithm of Beltrami and Indusi [2], which are randomized versions of the Pattern Search Method of Hooke and Jeeves [3]. In this paper a random search method with an optimized step-size is presented. The random vector is generated using a pseudorandom number generator with a normal distribution of zero mean and a given standard deviation.

2. Details of the method

Given a starting point X_0 , evaluate the function at this point. Generate a random vector which passes through X_0 and whose coordinates are drawn from a normal distribution of zero mean and a given standard deviation using subroutine RANDU (given in the appendix). Normalize this vector and call it R . The expression

* Presently with Mission Operation & Planning Division, ISRO Satellite Centre, Peenya, Bangalore 560058, INDIA.

$$X_i = X_0 + \lambda_i R \quad (1)$$

describes a point on the line passing through X_0 in the direction R . In particular, the points $X_1 = X_0 - R$, $X_2 = X_0$, $X_3 = X_0 + R$ correspond to the values $\lambda_1 = -1$, $\lambda_2 = 0$, $\lambda_3 = +1$.

Let $f_i = f(X_i)$. Then a , b , c may be determined so that the quadratic

$$F(\lambda) \equiv f(X_0 + \lambda R) = a\lambda^2 + b\lambda + c \quad (2)$$

passes through the three points $(f_1, -1)$, $(f_2, 0)$ and $(f_3, 1)$ in the (f, λ) plane. The minimum of $F(\lambda)$ on this curve exists if $F''(\lambda)$ is positive. If $F''(\lambda)$ is positive, then λ^* corresponding to the minimum of F is got by equating its first derivative to zero, i.e.

$$F'(\lambda)|_{\lambda=\lambda^*} = 2a\lambda^* + b = 0. \quad (3)$$

The coefficients a and b can be determined by solving the three linear equations

$$\begin{aligned} f_1 &= a(-1)^2 + b(-1) + c, \\ f_2 &= a(0)^2 + b(0) + c, \\ f_3 &= a(1)^2 + b(1) + c, \end{aligned} \quad (4)$$

which give

$$\begin{aligned} a &= \frac{1}{2}(f_1 - 2f_2 + f_3), \\ b &= (f_3 - f_1)/2. \end{aligned} \quad (5)$$

Substitution of (5) in (3) gives λ^* .

Now the function is evaluated at

$$X' = X_0 + \lambda^* R. \quad (6)$$

If $f(X') < f(X_0)$, then X_0 is replaced by X' and $f(X_0)$ by $f(X')$, and the search is continued from this point. Otherwise a new random direction passing through X_0 is generated, and the process is continued.

If $F''(\lambda)$ is negative, X_0 is replaced by the point yielding the least function value among X_i , $i = 1, 2, 3$, and then a new random direction is generated which passes through this point. This process is continued till the relative error falls below a prescribed positive number ϵ .

The various steps of the algorithm are given below:

Step 1: Let $f(X_0) = f_0$. Fix ϵ and IFIX. Set INDEX = 0, IEQUAL = 0, and initialize the random number generator.

Step 2: Generate a random vector R using subroutine RANDU and normalize it. Let this vector be R . Determine X_i , $i = 1, 2, 3$, using eq. (1) with $\lambda_i = -1, 0, 1$. Let the values of the function at X_1 and X_3 be f_1 and f_3 . Set $f_2 = f_0$ and INDEX = INDEX + 1.

- Step 3: Calculate $a = \frac{1}{2}(f_1 - 2f_2 + f_3)$.
 If $a > 0$, go to step 4.
 If $a \leq 0$, go to step 5.
- Step 4: Calculate $b = \frac{1}{2}(f_3 - f_1)$ and $\lambda^* = -b/2a$.
 Set $X' = X_0 + \lambda^*R$ and evaluate $f(X') = f'$.
 If $f' < f_0$, replace X_0 by X' and f_0 by f' , and go to step 6.
 If $f' \geq f_0$, go to step 2.
- Step 5: Let $f_0 = \min_{i=1,2,3} \{f_i\}$, let $X_0 =$ the corresponding X_i , and go to step 6.
- Step 6: If INDEX = 1, go to step 2.
 If INDEX > 1, test for convergence (i.e. test whether or not $|f_0^{(old)} - f_0^{(new)}| < \epsilon$).
 If the criterion is satisfied, stop;
 otherwise go to step 2.
 If $f_0^{(old)}$ is the same as $f_0^{(new)}$, set IEQUAL = IEQUAL + 1.
 If IEQUAL > IFIX, stop;
 otherwise go to step 2 to generate a new random direction.

Fig. 1 gives the flow chart for OSSRS.

3. Numerical examples

The algorithm was applied to the following test problems:

1. Rosenbrock's parabolic valley function

$$f(\mathbf{x}) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2.$$

The starting point is (-1.2, 1.0).

The minimum is 0 at (1, 1).

2. Rosenbrock's cubic function

$$f(\mathbf{x}) = 100(x_2 - x_1^3)^2 + (1 - x_1)^2.$$

The starting point is (-1.2, 1.0).

The minimum is 0 at (1, 1).

3. Beale's function

$$f(\mathbf{x}) = \sum_{i=1}^3 (C_i - x_1(1 - x_2^i))^2,$$

where $C_1 = 1.5$, $C_2 = 2.25$ and $C_3 = 2.625$.

The starting point is (0, 0).

The minimum is 0 at (3, 0.5).

4. Bigg's function

$$f(\mathbf{x}) = \sum_{i=1}^{10} [\exp(-x_1 y_i) - x_3 \exp(\{-0.1 i x_2 - y_i\}^2)],$$

where $y_i = \exp(-0.1i) - 5 \exp(-10\{0.1i\})$.

The starting point is (1, 2, 1).

The minimum is 0 at (1, 10, 5).

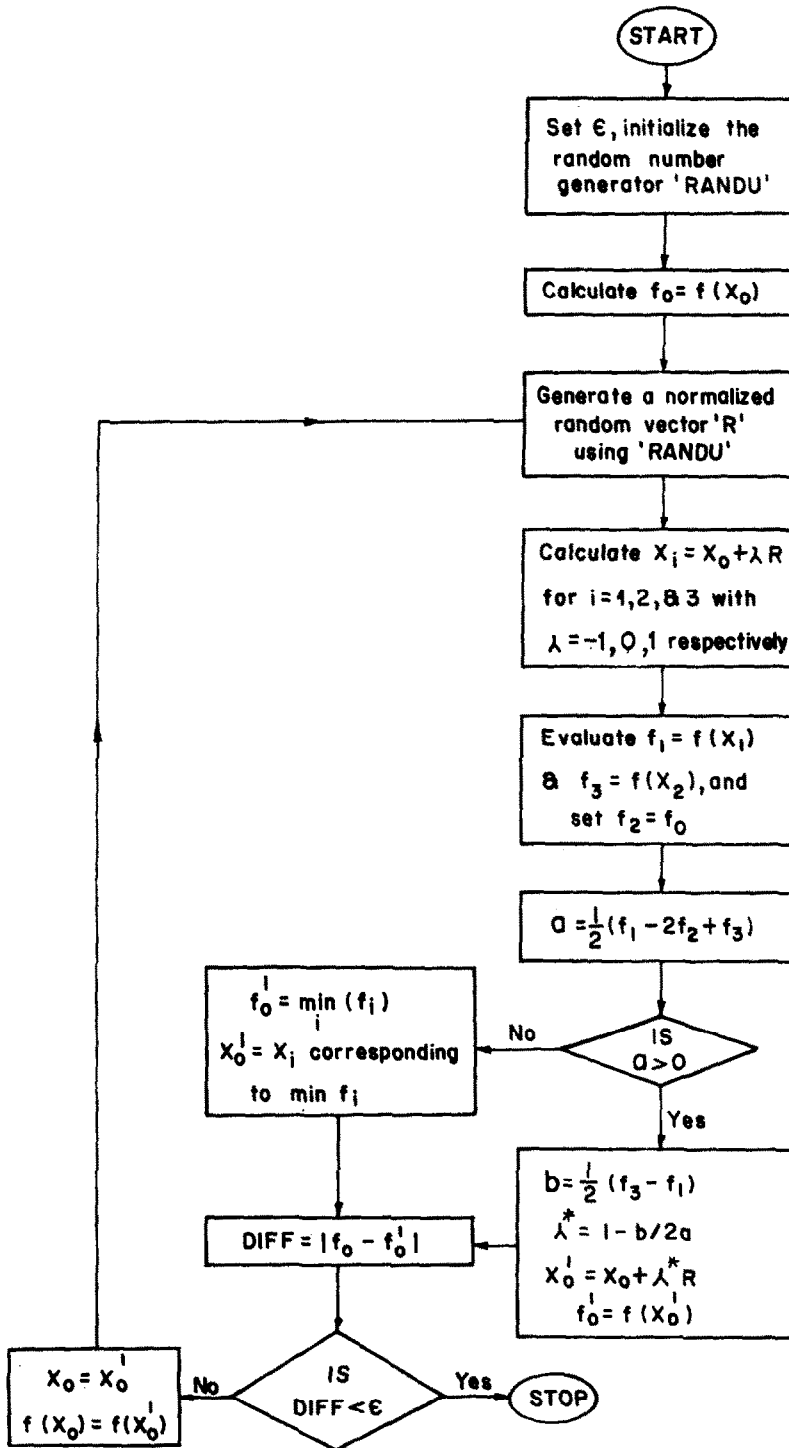


Fig. 1. Flow chart for OSSRS.

5. Powell's function

$$f(\mathbf{x}) = (x_1 + 10x_2)^2 + 5(x_3 - x_4)^2 + (x_2 - 2x_3)^4 + (10x_1 - x_4)^4 .$$

The starting point is (3, -1, 0, 1).

The minimum is 0 at (0, 0, 0, 0).

6. Colville's function

$$f(\mathbf{x}) = 100(x_1^2 - x_2)^2 + (1 - x_1)^2 + 10(x_4 - x_3^2)^2 + (1 - x_3)^2 + 10.1(x_2 - 1)^2 + (x_4 - 1)^2 + 19.8(x_2 - 1)(x_4 - 1) .$$

The starting point is (-3, -1, -3, -1).

The minimum is 0 at (1, 1, 1, 1).

Table 1.1. Problem 1

NFE	$f(\mathbf{x})$	x_1	x_2
1	24.1999	-1.2	1.0
81	0.2377	0.513	0.265
132	$.781 \times 10^{-1}$	0.792	0.641
318	0.413×10^{-4}	0.993	0.987
1941	0.657×10^{-6}	1.000	1.001

Table 1.2. Problem 2

NFE	$f(\mathbf{x})$	x_1	x_2
1	749.037	-1.2	1.0
8	6.662	0.715	0.623
11	0.654	0.654	0.366
175	0.41×10^{-3}	0.990	0.973
316	0.915×10^{-4}	0.990	0.971

Table 1.3. Problem 3

NFE	$f(\mathbf{x})$	x_1	x_2
1	14.203	0.0	0.0
3	4.481	0.9997	0.0247
8	0.167	2.336	0.257
56	0.571×10^{-1}	2.598	0.348
325	0.428×10^{-2}	3.113	0.536
373	0.124×10^{-3}	2.975	0.494
988	0.737×10^{-4}	2.978	0.494

Table 1.4. Problem 4

NFE	$f(x)$	x_1	x_2	x_3
1	1.598	1.0	2.0	1.0
39	0.922	0.400	2.135	1.498
247	0.903×10^{-1}	0.891	6.879	3.278
373	0.976×10^{-2}	0.972	8.848	4.358
677	0.932×10^{-3}	0.991	9.624	4.803
977	0.268×10^{-4}	0.997	9.970	4.973
1046	0.994×10^{-5}	0.998	9.971	4.979
1106	0.153×10^{-6}	1.000	10.002	5.001

Table 1.5. Problem 5

NFE	$f(x)$	x_1	x_2	x_3	x_4
1	707336	3.0	-1.0	0.0	1.0
5	55989.6	1.544	-0.733	-0.703	.066
9	1034.45	0.458	-1.484	-0.185	-0.770
11	535.63	0.291	-1.007	-0.198	-1.633
13	23.32	-0.316	-0.312	-0.581	-1.634
28	9.61	-0.094	-0.098	-0.329	-1.607
88	0.32	-0.049	-0.002	-0.306	-0.493
226	0.88×10^{-1}	-0.039	-0.001	-0.264	-0.228
451	0.77×10^{-2}	0.0037	-0.0041	-0.118	-0.14
4006	0.83×10^{-3}	-0.0211	0.0019	-0.045	-0.045

Table 1.6. Problem 6

NFE	$f(x)$	x_1	x_2	x_3	x_4
1	19192.0	-3.0	-1.0	-3.0	-1.0
3	7693.56	-2.197	-0.980	-2.459	-0.748
13	998.34	-0.108	-2.220	-1.393	+0.057
35	96.74	0.25	-0.54	-0.063	0.33
120	9.45	-0.52	0.33	-1.07	1.18
6295	0.97	1.375	1.875	0.305	0.095
25735	0.33×10^{-1}	1.04	1.11	0.93	0.87
30406	0.99×10^{-2}	1.04	1.09	0.94	0.89
97813	0.98×10^{-3}	1.01	1.02	0.98	0.97

Table 2. Comparison between uniform and normal distributions for problem 1

NFE	$f(x)$ (with unif. dist.)	$f(x)$ (with nor. dist.)
1	24.199	24.199
4	4.837	23.45
172	2.94	0.58×10^{-1}
318	0.289	0.41×10^{-4}
1941	0.60×10^{-5}	0.65×10^{-6}

Table 3. Number of function evaluations NFE vs. function value $f(\mathbf{x})$ for various values of σ for problem 2

NFE	$f(\mathbf{x})$				
	0.2	0.4	0.6	0.8	1.0
8	6.66215	6.66216	6.66215	6.66209	6.66211
11	0.854201	0.854201	0.854202	0.854210	0.854212
172	0.232773	0.232776	0.232767	0.232753	0.232766
316	0.915926×10^{-4}	0.915912×10^{-4}	0.916019×10^{-4}	0.916050×10^{-4}	0.915987×10^{-4}

4. Discussion of results

Tables 1.1–1.6 give the results for various test problems (NFE is the number of function evaluations). All computations were carried out on an IBM 360/44. In all these cases the standard deviation σ in RANDU was fixed at 0.2. Test problem 1 was solved using a uniform random number generator also; table 2 gives the comparison of results obtained using the two random number generators. Table 3 gives the result for test problem 2 for various values of σ – this shows that the convergence is not very sensitive to changes in σ . In the case of test problem 6 the convergence is very fast in the beginning and exceedingly slow near the vicinity of the optimum – this can be attributed to the narrow curved valleys exhibited by this function.

5. Conclusions

A new random search method with optimized step size is presented with results for various test problems. Efforts are on the way to incorporate constraints into the algorithm.

Acknowledgment

The author is grateful to the authorities of the National Aeronautical Laboratory, Bangalore, India for providing facilities to carry out this work. The author's thanks are also due to Miss B. Bharathi Devi, Structures Division, N.A.L, Bangalore, for providing her with the subroutine for uniform random number generator.

Appendix

```

SUBROUTINE RANDU (IX, IY, Y1, Y2)
COMMON/SHE/SIGMA
DIMENSION R(2)
AMEAN = 0
DO 1 I = 1, 2
IY = IX * 65539
IF (IY) 5, 6, 6

```

```
5  IY = IY + 2147483647 + 1
6  YFL = IY
   YFL = YFL * 0.4656613E - 09
   IX = IY
1  R(I) = YFL
   ARG = 2 * 3.141596 * R(2)
   CFT = SQRT (-2.0 * ALOG (R (1)))
   X1 = CFT * COS (ARG)
   X2 = CFT * SIN (ARG)
   Y1 = AMEAN + SIGMA * X1
   Y2 = AMEAN + SIGMA * X2
   RETURN
   END
```

References

- [1] J.P. Lawrence, III, and K. Steiglitz, Randomized pattern search, IEEE Trans. Comp. C-21 (1972) 382–385.
- [2] E.T. Beltrami and J.P. Indusi, An adaptive random search algorithm for constrained minimization, IEEE, Trans. Comps. C-21 (1972) 1004–1007.
- [3] R. Hooke and T.A. Jeeves, Direct search solution of numerical and statistical problems, J. ACM, 8 (1961) 212.