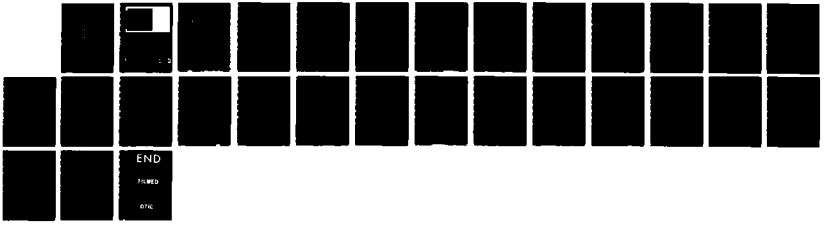
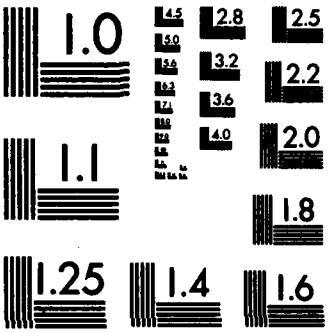
AD-A154 878 SIGMA - A STOCHASTIC-INTEGRATION GLOBAL MINIMIZATION ALGORITHM(U) WISCONSIN UNIV-MADISON MATHEMATICS RESEARCH CENTER F ALUFFI-PENTINI ET AL. MAR 85 1/1

UNCLASSIFIED MRC-TSR-2806 DAJA37-81-C-0740 F/G 9/2 NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

MRC Technical Summary Report #2806

SIGMA - A STOCHASTIC-INTEGRATION GLOBAL MINIMIZATION ALGORITHM

Filippo Aluffi-Pentini,
Valerio Parisi and
Francesco Zirilli

Mathematics Research Center
University of Wisconsin—Madison
610 Walnut Street
Madison, Wisconsin 53705

March 1985

(Received December 13, 1984)

Sponsored by

U. S. Army Research Office
P. O. Box 12211
Research Triangle Park
North Carolina 27709

UNIVERSITY OF WISCONSIN - MADISON
MATHEMATICS RESEARCH CENTER

SIGMA - A STOCHASTIC-INTEGRATION GLOBAL MINIMIZATION ALGORITHM

Filippo Aluffi-Pentini*,
Valerio Parisi** and Francesco Zirilli***

Technical Summary Report #2806

March 1985

ABSTRACT

The SIGMA package is a set of FORTRAN subprograms, using double-precision floating-point arithmetics, which attempts to find a global minimizer of a real-valued function $f(\underline{x}) = f(x_1, \ldots, x_N)$ of N real variables $x_1, \ldots, x_N$.

AMS (MOS) Subject Classifications: 65K10, 60H10, 49D25

cont → Key Words: Algorithms, Global Optimization, Stochastic Differential Equations.

Work Unit Number 5 - Optimization and Large Scale Systems

* Dipartimento di Matematica, Università di Bari, 70125 Bari (Italy).

** Istituto di Fisica, 2ª Università di Roma "Tor Vergata", Via Orazio Raimondo, 00173 (La Romanina) Roma (Italy).

*** Istituto di Matematica, Università di Salerno, 84100 Salerno (Italy).

Sponsored by the United States Army under Contract No. DAAG29-80-C-0041. The research reported in this paper has been made possible through the support and sponsorship of the U.S. Government through its European Research Office of the U.S. Army under Contract No. DAJA-37-81-C-0740 with the University of Camerino, Italy.

SIGNIFICANCE AND EXPLANATION

The paper gives a detailed description of a FORTRAN IV program based on a new method for finding a "global" (or "absolute") minimizer of a function of N real variables, i.e. the point x in N-dimensional space (or possibly one of the points) such that not only the function increases if one moves away from x in any direction, ("local" or "relative" minimum), but also such that no other point exists where f has a lower value.

The method, which was first proposed by the present authors in a paper which is to appear in the Journal of Optimization Theory and Applications, is based on ideas from statistical mechanics, and looks for a point of global minimum by following the solution trajectories of a stochastic differential equation representing the motion of particle (in N-space) under the action of a potential field and of a random perturbing force.

The complete algorithm is described in detail in the companion paper by the same authors: A Global Optimization Algorithm using Stochastic Differential Equations, which also contains a summary of the results of extensive numerical testing.

The tests were performed by running the program on an extensive set of carefully selected test problems of varying difficulty, and the performance was remarkably successful, even on very hard problems (e.g. problems with a single point of global minimum and up to about 10^{10} points of non-global minimum). →cont keywords include: see page-A- 10 to the 10th power

It is finally to be noted that the majority of the optimization methods presently available deal with the local optimization problem, and that no methods of comparable power seem to be available in the field of global optimization.

The program is essentially designed for portability on different types of computers, since it meets the specifications of PFORT, a portable subset of ANS FORTRAN developed at Bell Laboratories.

SIGMA - A STOCHASTIC-INTEGRATION GLOBAL MINIMIZATION ALGORITHM

Filippo Aluffi-Pentini*,
Valerio Parisi** and Francesco Zirilli***

1. METHOD

The algorithm used by SIGMA is described in detail in ref. [1].

A global minimizer of $f(\underline{x})$ is sought by monitoring the values of $f(\underline{x})$ along trajectories generated by a suitable discretization of the stochastic differential equation

$$d\underline{\xi} = -\nabla f(\underline{\xi})dt + \varepsilon(t)d\underline{w}$$

with initial condition:

$$\underline{\xi}(0) = \underline{x}_0$$

where ∇f is the gradient of f, $\underline{w}(t)$ is an N-dimensional standard Wiener process, and the "noise coefficient" $\varepsilon(t)$ is a positive function. The discretization has the form

$$\underline{\xi}_{k+1} = \underline{\xi}_k - h_k \underline{\tilde{Y}}(\underline{\xi}_k) + \varepsilon(t_k)\cdot\sqrt{h_k}\,\underline{u}_k, \qquad k = 0,1,2, \ldots$$

$$\underline{\xi}_0 = \underline{x}_0$$

where h_k is the time integration steplength, $\frac{1}{N}\underline{\tilde{Y}}(\underline{\xi}_k)$ is computed as a finite-differences approximation to the directional derivative of f in a randomly chosen direction, and $\underline{u}_k$ is a random sample from an N-dimensional standard gaussian distribution.

*Dipartimento di Matematica, Università di Bari, 70125 Bari (Italy).

**Istituto di Fisica, 2ª Università di Roma "Tor Vergata", Via Orazio Raimondo, 00173 (La Romanina) Roma (Italy).

***Istituto di Matematica, Università di Salerno, 84100 Salerno (Italy).

Sponsored by the United States Army under Contract No. DAAG29-80-C-0041. The research reported in this paper has been made possible through the support and sponsorship of the U.S. Government through its European Research Office of the U.S. Army under Contract No. DAJA-37-81-C-0740 with the University of Camerino, Italy.

We consider the simultaneous evolution of a number N_{TRAJ} of trajectories during an "observation period" having the duration of a given number N_{HP} of time integration steps, and within which the noise coefficient $\varepsilon(t)$ of each trajectory is kept at a constant value ε_p, while the steplength h_k and the spatial increment Δx_k for computing $\tilde{\underline{Y}}(\underline{\xi}_k)$ are automatically adjusted for each trajectory by the algorithm.

At the end of every observation period a comparison is made between the trajectories: one of the trajectories is discarded, all other trajectories are naturally continued in the next observation period, and one of them is selected for "branching", that is for generating also a second continuation trajectory which differs from the first one only in the starting values for ε_p and Δx_k, and is considered as having the same "past history" of the first.

The number of N_{TRAJ} of simultaneously evolving trajectories remains therefore unaffected, and the second continuation trajectory takes the place, from a program-implementation point of view, of the discarded trajectory.

The set of simultaneous trajectories is considered as a single trial, and the complete algorithm is a set of repeated trials. A single trial is stopped, at the end of an observation period, if a maximum given number N_{PMAX} of observation periods has been reached, or if all the final values of $f(\underline{x})$ (except for the discarded trajectory) are equal (within numerical tolerances, and possibly at different points $\underline{x}$) to their minimum value f_{TFMIN} ("uniform stop" at the level f_{TFMIN}). In the former case the trial is considered unsuccessful, while in the latter case a comparison is made between the common final function value f_{TFMIN}

and the current best minimum function value f_{OPT} found so far from algorithm start: if $f_{TFMIN} > f_{OPT}$ the trial is again considered unsuccessful; and if $f_{TFMIN} = f_{OPT}$ (within numerical tolerances) the trial is considered successful at the level f_{OPT}.

The trials are repeated with different operating conditions (initial point $\underline{x}_0$, maximum trial length N_{PMAX}, seed of the noise generator, policy for selecting the starting value for ε_p in the second continuation trajectory after branching, and trial-start value for ε_p) and the complete algorithm is stopped — at the end of a trial — if a given number N_{SUC} of uniform stops at the current f_{OPT} level has been obtained, or if a given maximum number N_{TRIAL} of trials has been reached: success of the algorithm is claimed if at least one uniform stop occurred at the final value of f_{OPT}.

2. DESCRIPTION OF THE PACKAGE

The algorithm used by SIGMA (see sect. 2 and ref. [1]) has been coded in the form of a set of FORTRAN subprograms, using double-precision floating-point arithmetics, which are described below.

2.1. Language

All the coding is written in FORTRAN IV and meets the specifications of PFORT, a portable subset of A.N.S. FORTRAN (ref. [2]). The FORTRAN implicit type definition for integers is used throughout; all non-integer variables are double precision.

2.2. Description of the Subprograms

The SIGMA package consists of a principal subroutine SIGMA, a set of 27 auxiliary subroutines, INIT, REINIT, TRIAL, GENEVA, PERIOD, BRASI, ORDER, COMPAS, STEP, SSTEP, NEWH, DERFOR, DERCEN, RCLOPT, STOOPT, RANGE, INISCA, NOSCA, SEGSCA, VARSCA, CUMSCA, ACTSCA, MOVSCA, UPDSCA, ALKNUT, GAUSRV, UNITRV; a set of 7 auxiliary functions, IPREC, IPRECE, FUNCTØ, ITOLCH, EIGSCA, CHAOS, UNIFRN; and a driver subroutine SIGMA1 calling SIGMA. The subprograms are described below, The user interested only in the use of SIGMA may jump to Section 3.

We may group the subprograms as follows.

a) Subprograms for the numerical integration: STEP, SSTEP, DERFOR, DERCEN, FUNCTØ, RANGE, NEWH. The value of the function $f(\underline{x})$ is computed — whenever required in the numerical integration process — by calling the function FUNCTØ. FUNCTØ rescales the variables by calling VARSCA (see d)), calls RANGE to take care of the cases where the current point $\underline{x}$ is

outside the admissible range ([1], sect. 3.2.11) calls the user-supplied function FUNCT (sect. 3.5.1) to compute $f(\underline{x})$, and possibly updates the best current function minimum f_{OPT} and the corresponding minimizer $\underline{x}_{OPT}$ by calling STOOPT (see c)). The basic step of the numerical integration is performed by SSTEP which calls FUNCTØ to compute the value of $f(\underline{x})$, and UNITRV (see e)) to compute the random direction along which the directional derivative is to be computed (see [1], sect. 3.2.2); the directional derivatives are computed numerically by SSTEP, with forward or central finite differences, by calling DERFOR or DERCEN, which call FUNCTØ; the first half-step ([1], sect. 3.2.1) is accepted or rejected ([1], sect. 3.2.3) by calling NEWH which also provides the updated value of the time integration steplength h_k; SSTEP also updates the cumulated scaling data ([1], sect. 3.2.12) by calling CUMSCA (see d)), and updates the spatial discretization increment Δx_k based on the results of calling ITOLCH. The second half-step ([1], sect. 3.2.1) is performed by SSTEP by calling GAUSRV (see e)) and can be accepted or rejected ([1], sect. 3.2.3). The subroutine STEP performs the single integration step for each one of the simultaneous trajectories by repeatedly calling SSTEP.

b) Subprograms for the selection of the trajectories: BRASI, ORDER, IPREC, IPRECE, COMPAS. The selection process for the trajectories ([1], sect. 3.2.6) is performed by the subroutine BRASI. BRASI first updates the trajectory data corresponding to the elapsed observation period, and then asks for an ordering of the trajectories by calling ORDER. ORDER obtains the ordering by comparing two trajectories on the basis of the past history, (by calling IPREC), and of the value of the noise coefficient ε_p (by calling IPRECE) ([1], sect. 3.2.6). Based on the ordering provided

by ORDER, BRASI

1) discards one of the trajectories

2) performs a branching on another trajectory, i.e. the trajectory to be branched gives rise to two "continuation" trajectories: the first one is unperturbed, and the second one has modified values for ε_p and for the initial Δx_k; the modified values are obtained from the old ones by means of random multiplicative factors which are computed with the aid of random number generator function CHAOS (see e)).

Since, from a program implementation point of view, the new trajectory is "moved" in the "position" of the discarded one, all the trajectory parameters must be moved to the new position. This is performed directly by BRASI for all the trajectory data, except for the scaling data which are moved by MOVSCA (see d)). Finally BRASI calls COMPAS in order to examine the stored data about past trajectories from the point of view of their utility to the only user of such data, which is the subroutine IPREC, and irrelevant data are discarded.

c) Subprograms for general management of the complete algorithm: SIGMA, INIT, REINIT, TRIAL, GENEVA, PERIOD, ITOLCH, RCLOPT, STOOPT.

GENEVA performs the generation of the set of trajectory segments corresponding to the current observation period and the final processing and evaluation of the trajectories. GENEVA first updates the scaling arrays containing A and $\underline{b}$ ([1], sect. 3.2.12) by calling SEGSCA and UPDSCA (see d)). The generation of the trajectory segments is performed by GENEVA by calling PERIOD.

PERIOD first computes the duration (in accepted steps) of the observation period, computes all the integration steps by repeatedly calling STEP (see a)) and finally performs the trajectory selection by calling BRASI (see b)).

Finally GENEVA determines some end-of-segment results (FPFMIN, FPFMAX, XPFMIN, see sect. 3.5.2) using the rescaling capabilities of SEGSCA and VARSCA (see d)).

The subroutine TRIAL generates a trial by repeatedly performing, for every observation period,

a call to GENEVA which generates the simultaneous trajectory segments, and performs the trajectory selection,

a (possible) call to PTSEG which performs end-of-segment output,

a check of the (trial) stopping criteria, with the aid of the function ITOLCH,

a decision about activating or deactivating the scaling of the variables (actions performed by calling ACTSCA or NOSCA).

The subroutine SIGMA is the principal subroutine of the package and is the only one which must be called by the user (apart from the driver SIGMA1, sect. 3.4).

SIGMA manages the execution of the complete algorithm, i.e. of a sequence of repeated trials performed by varying a number of operating conditions. SIGMA initializes the first trial by calling INIT, and the other trials by calling REINIT.

For each trial the subroutine SIGMA

enables or prevents a future activation (within the current trial) of the scaling of the variables by calling INISCA or NOSCA

actually executes the trial by calling TRIAL

updates a number of parameters (using ITOLCH and RCLOPT)

checks the algorithm-stopping criteria

possibly performs end-of-trial outputs by calling PTSEG, PTRIAL, and PTKSUC (see 3.5.2)

The subroutine STOOPT and RCLOPT respectively "store" and "recall" the current values of the best minimum FOPT and of the corresponding minimizer XOPT.

d) Subprograms for rescaling the variables: INISCA, NOSCA, SEGSCA, VARSCA, CUMSCA, ACTSCA, MOVSCA, UPDSCA, EIGSCA ([1], sect. 3.2.12).

INISCA initializes the common area /SCALE/ for the scaling data.

NOSCA deactivates the rescaling.

SEGSCA selects the trajectory which must be rescaled.

VARSCA computes the rescaled variables $A\underline{x} + \underline{b}$.

CUMSCA stores cumulated statistical data on the ill-conditioning of $f(A\underline{x} + \underline{b})$.

ACTSCA activates the rescaling.

MOVSCA moves the scaling data from the first to the second continuation of a branched trajectory.

UPDSCA updates the scaling matrix A and vector $\underline{b}$ by calling EIGSCA and VARSCA.

EIGSCA computes the largest eigenvalue of a matrix used for rescaling, starting from randomly chosen estimates (obtained by calling UNITRV) of the corresponding eigenvector.

e) Subprograms for pseudo-random number generation: CHAOS, UNIFRN, ALKNUT, GAUSRV, UNITRV.

CHAOS generates an element of a sequence of independent pseudo random numbers, each one having one out of four possible probability distributions (standard gaussian, standard Cauchy ([1], sect. 3.2.7), uniform in (-1,1) or (0,1)) by calling UNIFRN.

UNIFRN generates an element of a sequence of independent pseudo-random numbers uniformly distributed in (0,1), by calling ALKNUT and performing a further (nonlinear) randomization.

ALKNUT generates an element of a sequence of independent pseudo-random numbers (algorithm of Mitchell and Moore, modified as suggested by Brent, see ref. [3]).

GAUSRV generates an element of a sequence of independent pseudo-random N-vectors, having an N-dimensional standard gaussian probability distribution, by means of a rejection method, and based on uniformly distributed (-1,1) pseudo-random numbers obtained by calling CHAOS.

UNITRV generates an element of a sequence of independent pseudo-random N-vectors uniformly distributed on the unit sphere in R^N.

```
C
C MAIN PROGRAM (SAMPLE VERSION)
C CALLS SIGMA VIA THE DRIVER SUBROUTINE SIGMA1
C
      DOUBLE PRECISION  XØ, XMIN, FMIN
C
      DIMENSION XØ(2), XMIN(2)
C
C TEST PROBLEM DATA
C
C PROBLEM DIMENSION
      N = 2
C
C INITIAL POINT
      XØ(1) = Ø.DØ
      XØ(2) - Ø.DØ
C
C SET INPUT PARAMETERS
      NSUC = 3
      IPRINT = Ø
C
C CALL DRIVER SUBROUTINE SIGMA1
      CALL SIGMA1 (N,XØ,NSUC,INPRINT,XMIN,FMIN,NFEV,IOUT)
C
      STOP
      END
```

```
      DOUBLE PRECISION FUNCTION FUNCT (N,X)
C
C COMPUTES THE VALUE AT  X  OF THE SIX-HUMP CAMEL FUNCTION
C
      DOUBLE PRECISION X,XX,YY
      DIMENSION X(N)
      XX = X(1)*X(1)
      YY = X(2)*X(2)
      FUNCT = ((XX/3.DØ-2.1DØ)*XX+4.DØ)*XX+X(1)*X(2)
     1    + 4.DØ*(YY-1.DØ)*YY
      RETURN
      END
```

Figure 1 -- List of the Sample Program

3.6. Storage Requirements

The SIGMA package contains a total of about 1900 statements (including some 700 comment lines). This amounts on the ASCII FORTRAN compiler (with optimization option) of the UNIVAC EXEC 8 operating system to a storage requirement of about 4000 (36-bit) words for the instructions, about 3500 words for the data, and about 14,000 words for the COMMON area. The requirement for the array dimensions are 4N 36-bit words.

3.7. Example

Let $N = 2$, $\underline{x} = (x_1, x_2)^T$, and consider the six-humps camel function $f(\underline{x}) = \frac{1}{3} x_1^6 - 2.1\, x_1^4 + 4x_1^2 + x_1 x_2 + 4x_2^4 - 4x_2^2$ which has four non-global minima, and two global minima at $\underline{x} \simeq \pm (-0.089842,\ 0.71266)^T$ where $f \simeq -1.03163$. The sample program listed in fig. 1, which uses the easy-to-use driver SIGMA1, was run on a UNIVAC 1100/82 computer with EXEC8 operating system (level 38R5) and ASCII FORTRAN compiler (version 10R1A), starting from $\underline{x}_0 = (0,0)^T$ and with NSUC = 3.

The program claimed success (IOUT = 1) stopping correctly at one of the global minimizers, using 19660 function evaluations. The printout showed that if N_{SUC} had been equal to 1 (resp. 2), the minimum would have been found with only 2697 (resp. 8343) function evaluations.

ACKNOWLEDGEMENT: One of us (F.Z.) gratefully acknowledges the hospitality and the support of the Mathematics Research Center of the University of Wisconsin and of the Department of Mathematical Sciences of Rice University where part of this work was done.

|ISTOP| = 1 relative difference criterion satisfied

= 2 absolute difference criterion satisfied

= 3 both criteria satisfied

The sign of ISTOP indicates the relationship between the end-of-trial value FTFMIN and the best current minimum value FOPT (which is updated whenever a function value is computed).

ISTOP > 0 FTFMIN is numerically equal (with respect to at least one of the above difference criteria) to FOPT.

ISTOP < 0 FTFMIN is not even numerically equal to FOPT (and therefore cannot be considered an acceptable estimated global minimum).

ISTOPT is the value of the trial stopping indicator ISTOP corresponding to the (current or past) trial where FTFOPT was obtained, with the sign which is updated according to the comparison between FTFOPT and the present value of FOPT, as described above. The final value of ISTOPT is returned by SIGMA as the value of the output indicator IOUT (whenever the algorithm was started, IOUT ≠ -99, see above).

The subroutine definition statement of PTKSUC is

SUBROUTINE PTKSUC (KSUC)

where

KSUC is the integer variable (1 ≤ KSUC < NSUC)

defined above.

If IPRINT < 0 no calls are made to the output subroutines.

A user not interested in the use of any one of the output subroutines must provide the corresponding dummy subroutine (with RETURN as the only executable statement) in order to avoid unresolved references problems.

FOPT is the current best minimum value of f found from algorithm start (f_{OPT}) (FOPT is updated whenever a function value $f(\underline{x})$ is computed).

FTFMIN, FTFMAX are respectively the minimum and the maximum value of $f(\underline{x})$ among the end-of-trial values obtained at the final points of the last trajectories of the current trial (f_{TFMAX}, f_{TFMIN}).

FTFOPT is current minimum value of FTFMIN among the trials which did not stop due to the stopping condition related to NPMAX (stopping indicator ISTOP = 0, see below). FTFOPT is used by SIGMA to compute the input parameter KSUC for the subroutines PTKSUC, see below.

ISTOP is the indicator of the stopping condition of the trial, as follows:

ISTOP = 0 The maximum number NPMAX of observation periods has been reached.

ISTOP ≠ 0 all the final values of $f(\underline{x})$ of the last observation period (except for the just discarded trajectory) are close enough to their common minimum value FPFMIN, with respect to an absolute or relative difference criterion, ([1], sect. 3.2.13), to be considered numerically equal.

If ISTOP ≠ 0 the absolute value and the sign of ISTOP have the following meaning:

The absolute value indicates which of the difference criteria was satisfied

taken place, if NSUC (input parameter to SIGMA) had been given a (lower) value, equal to the current KSUC.

The subroutine PTKSUC is called only if IPRINT $\geq$ 0 and KSUC < NSUC.

The subroutine definition statement of PTSEG is

SUBROUTINE PTSEG (N, XPFMIN, FPFMIN, FPFMAX, KP, NFEV)

where

N is the dimension of the problem

FPFMIN and FPFMAX are respectively the minimum and the maximum value of $f(\underline{x})$ among the values obtained at the final points of the trajectory segments of the current observation period (excluding the discarded trajectory).

XPFMIN is the N-vector containing the coordinates of the final point (or possibly one of the points) where the function value FPFMIN was obtained.

KP is the total number of elapsed observation periods in the current trial.

NFEV is the total number of function evaluations performed from algorithm start.

The subroutine definition statement of PTRIAL is

SUBROUTINE PTRIAL (N, XOPT, FOPT, FTFMIN, FTFMAX, FTFOPT,
ISTOP, ISTOPT, NFEV, KP, IPRINT)

where

N is the dimension of the problem

XOPT is an N-vector containing the coordinates of the point (or possibly one of the points) where the current best minimum FOPT was obtained ($\underline{x}_{OPT}$).

alleviate the efficiency problems connected to the use of the explicit Euler step on ill-conditioned functions.

It is also recommended to avoid whenever possible to provide functions such that the "typical" values of the function and the coordinates (rough average values in the region of interest) differ from unity by too many orders of magnitude. Such a care is generally advisable due to some numerical values adopted in the FORTRAN implementation, for example to avoid overflow, but may be absolutely necessary when using the driver subroutine SIGMA1, due to the adopted general purpose default values for some input data, for example the stopping tolerances.

3.5.2. The Output Subroutines.

Apart from the output parameters in the call statement for SIGMA, the package is designed to be able to perform external output also by means of the calls to three output subroutines which must be supplied by the user: PTSEG, PTRIAL, and PTKSUC. The calls are activated according to the value of the control parameter IPRINT (sect. 3.2).

The subroutine PTSEG is called (if IPRINT > 0) at the end of every observation period.

The subroutine PTRIAL is called (if IPRINT ≥ 0) at the end of every trial.

The subroutine PTKSUC is called only at the end of every sucessful trial such that an increment occurred in the value KSUC of the maximum number of trials which had a uniform stop all at the same (current or past) value of f_{OPT}; a call to PTKSUC therefore provides the user with the operationally interesting information that a final success claim would have

3.5. User-supplied Subprograms.

The user must provide the function FUNCT which must compute the value of $f(\underline{x})$ (sect. 1), and the three output subroutines PTSEG, PTRIAL, PTKSUC. The above subprograms are described below: all non-integer arguments are double precision (integer arguments are indicated by means of the FORTRAN implicit type definition).

3.5.1. The function FUNCT

FUNCT must return as its value the value at $\underline{x}$ of the function f to be minimized.

The function definition statement is

DOUBLE PRECISION FUNCTION FUNCT (N,X)

where

N is the (input) dimension of the problem

X is the (input) N-vector containing the coordinates of the point $\underline{x}$ where the value of f is to be computed.

Note that the function $f(\underline{x})$ should comply with the growth conditions (2.3), (2.4) of [1], otherwise the function must be suitably modified; this may be performed by simply adding a penalization function, which must be zero on the region of interest. We note that this device can be used also to suitably restrict the search region (for example in the case of periodic functions).

It should be also noted that — although some form of automatic rescaling is provided by the algorithm — it is certainly advisable to avoid whenever possible to provide unnecessarily ill-conditioned functions (for example, due to careless choice of physical units), in order to

3.4 The Driver Subroutine SIGMA1

In order to both give an example of how to use SIGMA, and to save the average user the effort of deciding the numerical values for all the input parameters of SIGMA, a driver subroutine SIGMA1 is included in the package. SIGMA1 simply calls SIGMA after assigning default values to a number of input parameters. The subroutine definition statement is:

SUBROUTINE SIGMA1 (N, XØ, NSUC, IPRINT, XMIN, FMIN, NFEV, IOUT)

where the parameters have the same meaning as in SIGMA.

All the other input parameters of SIGMA are assigned default values within SIGMA1 as follows:

$H = 10^{-10}$

EPS = 1

$DX = 10^{-9}$

IRAND = 0

NTRAJ = 0

ISEGBR = 0

KPBRØ = 0

INKPBR = 0

NPMIN = 10

NPMAXØ = 100

INPMAX = 50

NTRIAL = max(50,5•NSUC)

$TOLREL = 10^{-3}$

$TOLABS = 10^{-6}$

KPASCA = 10 (if $N \leq 5$); = 300 (if $N > 5$)

INHP = 1

$XRMIN(I) = -10^{4}$ (I = 1,...,N)

$XRMAX(I) = 10^{4}$ (I = 1,...,N)

NPMIN (say 5 < NPMIN < 100)

NPMAXØ (say 0 < NPMAXØ < 150)

NTRIAL (say NTRIAL > 50 and NTRIAL > 5·NSUC)

INPMAX (say 30 < INPMAX < 100)

The following parameters have a marked effect on package performance and computational effort:

INHP, NTRAJ, ISEGBR, INKPBR, NSUC.

The magnitude of the effect roughly decreases from left to right.

In order to avoid untolerable growth of the computation effort or an unacceptable degradation of the performance, the user is advised to modify (if needed) the above parameters starting from NSUC, based on information from the output subroutines (PTRIAL and PTKSUC). Note that NSUC is the only "free" control parameter of the driver SIGMA1 (Sect. .4).

The value of KPASCA should be based on possible analytical or experimental evidence on the ill-scaling of the function $f(\underline{x})$. Choose a small value (say 10) for a badly scaled function, a large value (say 300) for a very well scaled function. The N-dimensional interval (XRMIN, XRMAX) should be as large as possible, consistently with the purpose of avoiding computation failures (e.g. overflow). Finally we note that due to the joint operation of the stopping conditions for the trial (see [1], sect. 3.2.8), in order to use only one of the conditions it is sufficient to put to zero the threshold tolerance (TOLREL or TOLABS) of the other condition. Suggested default values of most input parameters are provided in the driver subroutine SIGMA1 (sect. 3.4).

XMIN is an output N-vector containing the coordinates of the point (or possibly one of the points) where the final value FMIN of f_{OPT} was found.

FMIN is the final value of the best current minimum function value f_{OPT}.

NFEV is the (output) total number of function evaluations (including those used for the computation of derivatives, and for the rejected time-integration steps).

IOUT is the (output) indicator of the stopping conditions as follows: If IOUT = -99 a fatal error was detected when performing some preliminary checking of the input data, and the algorithm was not even started; otherwise the algorithm was started, and the value of IOUT is the final value of the of the parameter ISTOPT (an output indicator of the output subroutine PTRIAL, described in sect. 3.5.2.).

Success is claimed by the algorithm it IOUT > 0, i.e. if at least one of the trials stopped with a positive value of the trial stopping indicator ISTOP (described in sect. 3.5.2) and no lower value for f_{OPT} was found in the following trials.

3.3. Some Guidelines for the Choice of the Input Parameters.

Proper operation of the package should be almost independent of IRAND, KPBRØ (and XØ). The performance of the package should not be too sensitive to H, EPS, DX, since these are initial values of variables which are adaptively controlled by the program.

The following parameters are expected to have little effect on the performance, as long as they belong to wide "insensitivity" bounds:

XRMIN, XRMAX are input N-vectors defining an admissible region for the x-values, within which the function values can be safely computed (see [1], sect. 3.2.11, where XRMIN(I), XRMAX(I) are called R_i^{MIN}, R_i^{MAX}).

KPASCA is the (input) minimum number of observation periods, before the scaling procedures are activated (K_{pasca}).

IRAND is a control (input) index for the initialization of the random number generator:

if IRAND > 0 the generator is initialized before starting the trial K_t with seed IRAND + K_t-1;

if IRAND ≤ 0 the generator is initialized (with seed 0) only at the first call of SIGMA.

INHP is used to control the number NHP ("duration") of time integration steps for observation period K_p as follows:

if INHP = 1, NHP = 1 + $[\log_2(K_p)]$, ("short" duration)

if INHP = 2, NHP = $[\sqrt{K_p}]$ ("medium" duration)

if INHP = 3, NHP = K_p ("long" duration),

where [x] is the greatest integer not greater than x.

IPRINT is an input control index used to control the amount of printed output by controlling the calls to the user-supplied output subroutines PTSEG (end-of-segment output), PTRIAL (end-of-trial output), and PTKSUC (end-of-trial output related to the count of successful trials), described in sect. 3.5.2.

if IPRINT < 0 no call to the print subroutines

if IPRINT = 0 call only PTRIAL and PTKSUC

if IPRINT > 0 all the print subroutines are called.

to a default value (NTRAJ = 7), and if the input value is outside the interval (3,20) NTRAJ is set to the nearest extreme value).

ISEGBR, KPBRØ, INKPBR are the parameters I_b, K_{po}, M_p which determine which one of the simultaneous trajectories is to be branched (see [1], sect. 3.2.6). (Note however that if one of the input values is zero, the corresponding variable is set to a default value: ISEGBR = (1+NTRAJ)/2, (FORTRAN integer division), INKPBR = 10, KPBRØ = 3; if the input value for ISEGBR is outside the interval (1,NTRAJ), ISEBGR is set to the nearest extreme value; and if KPBRØ has a value not inside the interval (1,INKPBR), it is assigned the same value modulo INKPBR).

NPMIN is the (input) minimum duration of a trial, i.e. the minimum number of observation periods before checking the trial stopping condition.

NPMAXØ is the (input) initial value (i.e. for the first trial) for the maximum duration of a trial, i.e. for the maximum acceptable number NPMAX of observation periods in a trial (N_{PMAX}).

INPMAX is the (input) increment for NPMAX, when NPMAX is varied from one trial to the following one.

NSUC is the (input) number of successful trials (with the same final value f_{OPT}, see sect. 1) after which the computation is stopped (N_{SUC}).

NTRIAL is the (input) maximum allowed number of trials, after which the computation is stopped (N_{TRIAL}).

TOLREL and TOLABS are the (input) relative and absolute tolerances for stopping a single trial (τ_{REL}, τ_{ABS}, see [1], sect. 3.2.8).

N, XØ, H, EPS, DX,

NTRAJ, ISEGBR, KPBRØ, INKPBR,

NPMIN, NPMAXØ, INPMAX,

NSUC, NTRIAL, TOLREL, TOLABS, XRMIN, XRMAX

KPASCA, IRAND, INHP, IPRINT

SIGMA returns to the calling program the output parameters

NTRAJ, ISEGBR, KPBRØ, INKPBR,

XMIN, FMIN, NFEV, IOUT

The call parameters are described in the next section.

We note that the SIGMA package gives the user the possibility of obtaining — during algorithm evolution — the values of a number of other parameters by means of the output subroutine (to be supplied by the user) which are described in sect. 3.5.2. The parameters are

KP, NF, XOPT, FOPT

XPFMIN, FPFMIN, FPFMAX, FTFMIN, FTFMAX, FTFOPT

ISTOP, ISTOPT, KSUC

and are described in sect. 3.5.2.

3.2. Description of the parameters of the call statement for SIGMA.

N is the problem dimension (number of coordinates of a point $\underline{x}$)

XØ is an N-vector containing the initial values of the x-variables

H is the initial value of the time integration steplength.

EPS is the initial value of the noise coefficient

DX initial value of the magnitude of the discretization increment (Δx) for computing the finite-differences derivatives.

NTRAJ is the number of simultaneous trajectory segments (N_{TRAJ}). (Note however that if the input value is zero, NTRAJ is set

3. USAGE

In order to use the package the user must provide:

a) a driver program which calls the principal subroutine SIGMA,

b) a set of four auxiliary subprograms (to compute the function $f(\underline{x})$ and to output the results).

The CALL statement for SIGMA is described in sect. 3.1, the parameters of the CALL statement are described in sect. 3.2. Some guidelines for the choice of the values of the input parameters are given in sect. 3.3. A sample driver subroutine (SIGMA1) which calls SIGMA is described in sect. 3.4: such a subroutine assigns default values to a number of input parameters to SIGMA: it has therefore a considerably lower number of input parameters, and can be used as an easy-to-use driver for the average user. The user-supplied subprograms are described in sect. 3.5.

3.1. Call to SIGMA

The call statement is

```
CALL SIGMA (N, XØ, H, EPS, DX,
            NTRAJ, ISEGBR, KPBRØ, INKPBR,
            NPMIN, NPMAXØ, INPMAX,
            NSUC, NTRIAL, TOLREL, TOLABS, XRMIN, XRMAX,
            KPASCA, IRAND, INHP, IPRINT,
            XMIN, FMIN, NFEV, IOUT)
```

The program calling SIGMA must set the input call parameters

REFERENCES

[1] Aluffi-Pentini F., Parisi V., and Zirilli F.: A global optimization algorithm using stochastic differential quations, ACM T.O.M.S (this issue).

[2] Ryder B. G., and Hall A.D.: The PFORT verifier. Computing Science Tcchnical Report n. 12, I.T.T. Bell Laboratories, Murray Hill, N.J., Jan. 1981.

[3] Knuth D.E.: The art of computer programming, vol. II Seminumerical algorithms, 2nd edition, Addison-Wesley, Reading, Mass., 1981, p. 26-28.

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER #2806	2. GOVT ACCESSION NO. AD-A154878	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) SIGMA - A STOCHASTIC-INTEGRATION GLOBAL MINIMIZATION ALGORITHM		5. TYPE OF REPORT & PERIOD COVERED Summary Report - no specific reporting period 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Filippo Aluffi-Pentini, Valerio Parisi and Francesco Zirilli		8. CONTRACT OR GRANT NUMBER(s) DAJA-37-81-C-0740 DAAG29-80-C-0041
9. PERFORMING ORGANIZATION NAME AND ADDRESS Mathematics Research Center, University of Wisconsin 610 Walnut Street Madison, Wisconsin 53705		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS Work Unit Number 5 - Optimization and Large Scale Systems
11. CONTROLLING OFFICE NAME AND ADDRESS U. S. Army Research Office P.O. Box 12211 Research Triangle Park, North Carolina 27709		12. REPORT DATE March 1985 13. NUMBER OF PAGES 24
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASSIFIED 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Algorithms, Global Optimization, Stochastic Differential Equations

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

The SIGMA package is a set of FORTRAN subprograms, using double-precision floating-point arithmetics, which attempts to find a global minimizer of a real-valued function $f(\underline{x}) = f(x_1, \ldots, x_N)$ of N real variables $x_1, \ldots, x_N$.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

END

FILMED

7-85

DTIC