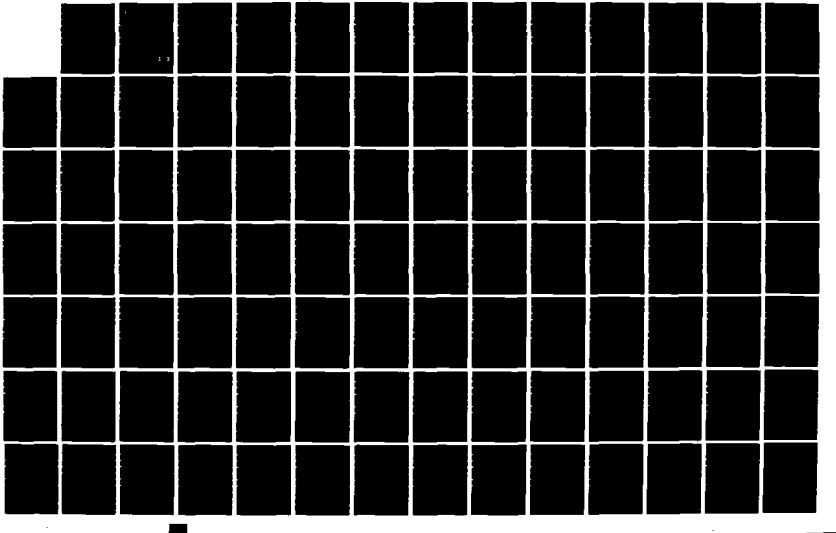
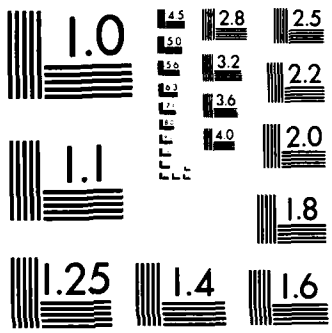
AD-A157 546 A NEW METHOD FOR GLOBAL OPTIMIZATION BASED ON STOCHASTIC DIFFERENTIAL EQUATIONS(U) CAMERINO UNIV (ITALY) MATHEMATICS INST F ALUFFI-PENTINI ET AL. 1/3

UNCLASSIFIED DEC 84 DAJA37-81-C-0740 F/G 12/1 NL





MICROCOPY RESOLUTION TEST CHART

)ARDS-1963-A

(2)

AD

A NEW METHOD FOR GLOBAL OPTIMIZATION BASED ON STOCHASTIC DIFFERENTIAL EQUATIONS

Final Technical Report

by

Filippo Aluffi - Pentini
Valerio Parisi
Francesco Zirilli

December 1984

United States Army
EUROPEAN RESEARCH OFFICE OF THE U. S. ARMY
London England

CONTRACT NUMBER DAJA 37-81-C-0740
Università di Camerino, Italy

Approved for Public Release; distribution unlimited

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO AD-A157546	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) A NEW METHOD FOR GLOBAL OPTIMIZATION BASED ON STOCHASTIC DIFFERENTIAL EQUATIONS		5. TYPE OF REPORT & PERIOD COVERED Final Technical 12.81-12.84
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Filippo Aluffi-Pentini, Valerio Parisi, Francesco Zirilli		8. CONTRACT OR GRANT NUMBER(s) DAJA37-81-C-0740
9. PERFORMING ORGANIZATION NAME AND ADDRESS Università di Camerino, Istituto Matematico - Via V. Venanzi 2 - Attn. Prof F. Zirilli I-62033 Camerino (MC) Italy		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 6.11.02A 1T161102BH57-05
11. CONTROLLING OFFICE NAME AND ADDRESS CDR, USARDSG-UK, Box 65, FPO NY 09510		12. REPORT DATE December 1984
		13. NUMBER OF PAGES 300
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 is 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)

Global optimization, Stochastic differential equations
Numerical Analysis
Mathematical software
Algorithm analysis, certification and testing

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

A new approach is presented to the problem of finding a global (i.e. absolute) minimizer of a function of several real variables, and some of its mathematical properties are investigated.

The approach is based on the idea of following the solution trajectories of a stochastic differential equation inspired by statistical mechanics.

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

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

We also describe a complete algorithm (SIGMA) based on the above approach, which looks for a point of global minimum by monitoring the values of the function to be minimized along a number of simultaneously-evolving trajectories generated by a new (stochastic) scheme for the numerical integration of the stochastic differential equation.

Finally we describe the software package SIGMA which implements the above algorithm in a portable subset of the A.N.S. FORTRAN IV language, a number of carefully selected test problems designed for testing the software for global optimization, and the results of testing SIGMA on the above problems, and on a problem of theoretical chemistry.

The main conclusion is that the performance of SIGMA is very good, even on some very hard problems.

AD

A NEW METHOD FOR GLOBAL OPTIMIZATION BASED ON STOCHASTIC DIFFERENTIAL EQUATIONS

Final Technical Report

by

Filippo Aluffi-Pentini

Valerio Parisi

Francesco Zirilli

December 1984

United States Army

EUROPEAN RESEARCH OFFICE OF THE U.S. ARMY

London England

CONTRACT NUMBER DAJA 37-81-C-0740

Università di Camerino, Italy

Abstract

A new approach is presented to the problem of finding a global (i.e. absolute) minimizer of a function of several real variables, and some of its mathematical properties are investigated.

The approach is based on the idea of following the solution trajectories of a stochastic differential equation inspired by statistical mechanics.

We also describe a complete algorithm (SIGMA) based on the above approach, which looks for a point of global minimum by monitoring the values of the function to be minimized along a number of simultaneously-evolving trajectories generated by a new (stochastic) scheme for the numerical integration of the stochastic differential equation.

Finally we describe the software package SIGMA which implements the above algorithm in a portable subset of the A.N.S. FORTRAN IV language, a number of carefully selected test problems designed for testing the software for global optimization, and the results of testing SIGMA on the above problems, and on a problem of theoretical chemistry.

The main conclusion is that the performance of SIGMA is very good, even on some very hard problems.

Keywords

Global optimization
Stochastic differential equations
Numerical Analysis
Mathematical software
Algorithm analysis, certification and testing.

Table of Contents

List of appendices

1. Introduction

This is the final report on the work done from December 1981 to December 1984, under contract n. DAJA 37-81-C-0740 awarded to Università di Camerino, Italy, on the research project "Numerical Optimization", by the principal investigator Francesco Zirilli and his co-workers.

The objective of the research was to develop a new method for global optimization, founded on a stochastic differential equation obtained by means of a time-dependent stochastic perturbation of an ordinary differential equation.

This included working on the mathematical foundations of the method, and building up a robust numerical algorithm for global optimization.

The research has produced:

- The development of a robust numerical algorithm for global optimization, the algorithm SIGMA.
- Studies on the mathematical foundations of the method.
- An extensively tested and well-performing FORTRAN implementation of the algorithm, the software package SIGMA.
- The development of a set of carefully selected problems to be used for testing global optimization software.
- Two FORTRAN subroutines implementing the above set of test problems.
- A successful application of the algorithm to a problem in theoretical chemistry.

The research has also stimulated scientific contacts with several italian and foreign scholars.

The results of the research have been disseminated by means of

- Six research papers submitted to high-standard professional or academic journals (three of them already accepted for publication).
- Short communications on the work in progress in national and international scientific meetings in Rome, Bonn, Milan, Bologna.
- Seminars at the University of L'Aquila, University of Salerno, Rice University (Houston, Texas), and Fondazione Donegani, Milan.

2. The proposed method for global cptimization

We consider the problem of finding a global minimizer of a given real-valued function f of N real variables $x_1, x_2, \ldots, x_N$, i.e. the point $\underline{x}^* = (x_1^*, \ldots, x_N^*)$ in the N-dimensional real euclidean space $\mathbb{R}^N$ such that f attains at $\underline{x}^*$ a global (or "absolute") minimum, defined by

$$(1) \qquad f(\underline{x}^*) \leqslant f(\underline{x}) \qquad \text{for all} \quad \underline{x} = (x_1, \ldots, x_N) \in \mathbb{R}^N$$

We assume that the function f is sufficiently regular, that its minimizers are isolated and non-degenerate, and that (for reasons that will become clear later)

$$(2) \qquad \lim_{\|\underline{x}\|_2 \to \infty} f(\underline{x}) = +\infty$$

in such a way that

$$(3) \qquad \int_{\mathbb{R}^N} \exp(-2f(\underline{x})/\varepsilon^2)\, d\underline{x} < +\infty$$

for all real $\varepsilon \neq 0$.

The interest of the global optimization problem both in mathematics and in many applications is well known and will not be discussed here.

We want just to remark here that the root-finding problem for the system $\underline{g}(\underline{x}) = \underline{0}$, where $\underline{g}: \mathbb{R}^N \to \mathbb{R}^N$ can be formulated as a global optimization problem considering the function $F(\underline{x}) = \|\underline{g}(\underline{x})\|_2^2$, where $\|\cdot\|_2$ is the euclidean norm in $\mathbb{R}^N$.

Despite its importance and the efforts of many researchers the global optimization problem is still rather open and there is a need for methods with solid mathematical foundation and good numerical performance.

Much more satisfactory is the situation for the problem of finding the local minimizers of f, where a large body of theoretical and numerical results exists.

Ordinary differential equations have been used in the study of the local optimization problem or of the root finding problem by several authors. The above methods usually obtain the local mi-

nimizers or roots by following the trajectories of suitable ordinary differential equations.

The simplest example is the first-order "steepest descent" equation

$$\frac{d\underline{x}}{dt} = - \nabla f(\underline{x}) \tag{4}$$

where ∇f is the gradient of the function f to be minimized.

However, since the property of being a global minimizer is a global one, that is, depends on the behaviour of f at each point of $\mathbb{R}^N$, and the methods that follow a trajectory of an ordinary differential equation are local, that is, they depend only on the behaviour of f along the trajectory, there is no hope of building a completely satisfactory method for global optimization based on ordinary differential equations.

The situation is different if we consider a suitable stochastic perturbation of an ordinary differential equation.

If we perturb the steepest-descent differential equation (4) by adding a "white-noise" term, we are led to consider the (Ito) stochastic differential equation

$$d\underline{\xi} = -\nabla f(\underline{\xi})\, dt + \varepsilon\, d\underline{w} \tag{5}$$

where ∇f is the gradient of the function f to be minimized, $\underline{w}(t)$ is a standard N-dimensional Wiener process ("Brownian motion"), and ε is a real "noise" coefficient.

Such equation is known as the Smoluchowski-Kramers equation, and can be considered as a singular limit of the second-order Langevin equation. when the inertial (i.e. second-order) term is neglected.

The Smoluchowski-Kramers equation has been extensively used by solid-state physicists and chemists to study physical phenomena such as atomic diffusion in crystals or chemical reactions.

In these applications eq. (5) represents diffusion across potential barriers under the stochastic forces $\varepsilon\, d\underline{w}$, where $\varepsilon = (\frac{2kT}{m})^{\frac{1}{2}}$, T is the absolute temperature, k the Boltzmann constant, m a suitable mass coefficient, and f is the potential energy.

The use of the above equation is suggested by the behaviour, for constant ε, of the stochastic process $\underline{\xi}(t)$, solution of the equation starting from an initial point $\underline{x}_0$.

It is well known that the probability density function

$p_\varepsilon(x,t)$ of the (random) value at time t of the solution process tends, as $t \to \infty$ (if condition (3) holds), to a limit "equilibrium" density

$$(6) \qquad p_\varepsilon(\underline{x}) = A_\varepsilon \; e^{-(2/\varepsilon^2)f(\underline{x})}$$

where A_ε is a normalization constant.

The equilibrium density is independent of the starting point $\underline{x}_0$ and is peaked at the global minimizers of f, with narrower peaks if ε is smaller.

In physical terms this indicates a greater concentration of particles at lower temperatures around the global minima of the potential energy.

Moreover in the limit $\varepsilon \to 0$ the equilibrium density becomes a weighted sum of Dirac's deltas concentrated at the global minimizers of f.

In order to obtain the global minimizers of f as asymptotic values as $t \to \infty$ of a sample trajectory of a suitable stochastic differential equation it seems natural to try to perform the limit $t \to \infty$ and the limit $\varepsilon \to 0$ together. We therefore consider the equation (5) with time-varying ε, that is

$$(7) \qquad d\underline{\xi} = -\nabla f(\underline{\xi})dt + \varepsilon(t)d\underline{w}$$

with initial condition

$$(8) \qquad \underline{\xi}(0) = \underline{x}_0$$

where

$$(9) \qquad \lim_{t \to \infty} \varepsilon(t) = 0.$$

In physical terms condition (9) means that the temperature T is decreased to absolute zero when $t \to \infty$, that is, the system is "frozen".

Since we want to end up in a global minimizer of f, that is, a global minimizer of the (potential) energy, the system has to be frozen very slowly (adiabatically).

Several mathematical questions related to the solutions of eqs. (5) and (7), such as the way in which $p_\varepsilon(\underline{x},t)$ approaches $p_\varepsilon(x)$ for a class of one-dimensional systems, or the rate at which $\varepsilon(t)$ should go to 0 in eq. (9), are considered in Appen-

dices 1 and 2.

The method we propose looks for a global minimizer of f by monitoring the values of f along a number of simultaneously-evolving numerical solution trajectories of the Cauchy problem (7), (8), which are generated by a new (stochastic) numerical-integration scheme.

3. The algorithm SIGMA

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

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

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

where h_k is the time integration steplength, $\frac{1}{N}\tilde{\underline{\gamma}}(\underline{\xi}_k)$ 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.

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 the 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 in-

Clearly $p_\infty^{\varepsilon_0}$ is the probability density of a random variable $\xi_\infty^{\varepsilon_0}$ so that $\xi^{\varepsilon_0}(t) \to \xi_\infty^{\varepsilon_0}$ in law when $t \to \infty$. Let us remark that $p_\infty^{\varepsilon_0}$ does not depend on the initial condition x_0.

We want to study the behaviour of $p_\infty^{\varepsilon_0}$ as $\varepsilon_0 \to 0$ and the rate of approach of p^{ε_0} to $p_\infty^{\varepsilon_0}$ as $t \to \infty$. We will consider for the sake of simplicity only the one-dimensional case when f is as in Fig. 1, i.e. with three extrema at the points $x_- < x_0 < x_+$, decreasing in $(-\infty, x_-)$ and (x_0, x_+) and increasing in (x_-, x_0) and $(x_+, +\infty)$, with $f(x) \to +\infty$ as $|x| \to \infty$ in such a way to satisfy (11) for all $\varepsilon_0 \neq 0$.

We have

$$\frac{df}{dx}(x_+) = \frac{df}{dx}(x_-) = \frac{df}{dx}(x_0) = 0 .$$

Using the following notation

$$f_+ = f(x_+) \qquad c_+ = \frac{d^2 f}{dx^2}(x_+)$$

$$f_- = f(x_-) \qquad c_- = \frac{d^2 f}{dx^2}(x_-)$$

$$f_0 = f(x_0) \qquad c_0 = -\frac{d^2 f}{dx^2}(x_0)$$

$$\Delta f_- = f_0 - f_- > 0 \qquad \Delta f_+ = f_0 - f_+ > 0$$

it is easy to prove the following result.

<u>Proposition 2.1.</u> Let f be as above and let c_0, c_+, c_- be greater than zero; then

i) If $\Delta f_- > \Delta f_+$ and $\exists$ [illegible] > 0 such that $f(x) \geq$ [illegible] $(x-x_-)^2 + f_-$ $\forall x \in \mathbb{R}$

then

Let $\xi^{\varepsilon_0}(t)$ be the stochastic process solution of (5), (6); for any Borel set $A \subset \mathbb{R}^n$ we define

$$P^{\varepsilon_0}(0,x_0,t,A,) = \mathbb{P}\{\xi^{\varepsilon_0}(t) \in A\} \tag{7}$$

where $\mathbb{P}\{\cdot\}$ is the probability of $\{\cdot\}$, and $P^{\varepsilon_0}(0,x_0,t,A)$ is the transition probability of $\xi^{\varepsilon_0}(t)$. Under regularity assumptions for f we have

$$P^{\varepsilon_0}(0,x_0,t,A) = \int_A p^{\varepsilon_0}(0,x_0,t,x)\,dx \tag{8}$$

where the transition probability density $p = p^{\varepsilon_0}(0,x_0,t,x)$ satisfies the following Fokker-Planck equation:

$$\frac{\partial p}{\partial t} = \frac{\varepsilon_0^2}{2}\,\Delta p + \operatorname{div}(\nabla f\, p) \tag{9}$$

with

$$\lim_{t\to 0} p^{\varepsilon_0}(0,x_0,t,x) = \delta(x - x_0) \tag{10}$$

where Δ and div are the laplacian and the divergence with respect to x and $\delta(\cdot)$ is the Dirac delta function.

Let A_{ε_0} be defined by

$$1/A_{\varepsilon_0} \equiv \int_{\mathbb{R}^n} e^{-2f(x)/\varepsilon_0^2}\,dx < \infty \tag{11}$$

then as $t \to \infty$ the transition probability density $p^{\varepsilon_0}(0,x_0,t,x)$ approaches the function

$$p_\infty^{\varepsilon_0}(0,x_0,x) = A_{\varepsilon_0}\, e^{-2f(x)/\varepsilon_0^2} \tag{12}$$

2. Method

Let us consider the Cauchy problem

$$d\xi = -\nabla f(\xi)\, dt + \varepsilon(t)\, dw \tag{3}$$

$$\xi(0) = x_0 \tag{4}$$

for the (Ito) stochastic differential equation (3), where $f:\mathbb{R}^n \to \mathbb{R}$ is the function to be (globally) minimized, ∇f is the gradient of f, $w(t)$ is an n-dimensional standardized Wiener process, and $\varepsilon(t)$ is a given function. We assume that f and ε are sufficiently well-behaved so that our statements are meaningful; in particular we assume that

$$\lim_{\|x\|_2 \to \infty} f(x) = +\infty$$

and

$$\int_{\mathbb{R}^n} e^{-\alpha f(x)}\, dx < \infty \qquad \forall \alpha \in \mathbb{R}\setminus\{0\}.$$

and that f has only a finite number of isolated global minimizers.

We propose to numerically integrate problem (3), (4) looking at the asymptotic value of a sample numerical trajectory solution to obtain a global minimizer of f. Let us start by considering the problem (3), (4) when $\varepsilon(t) = \varepsilon_0$ is a constant; that is

$$d\xi = -\nabla f(\xi)dt + \varepsilon_0 dw(t) \tag{5}$$

$$\xi(0) = x_0.$$

minimizers of f, with narrower peaks if the constant ε_0 is smaller.

The method we propose attempts to obtain a global minimizer of f by looking at the asymptotic value, as $t \to \infty$, of a numerically computed sample trajectory of an equation like (2) where ε is a function of time $\varepsilon(t)$ which tend to zero in a suitable way as $t \to \infty$. Similar ideas in the context of discrete optimization have been introduced by Kirkpatrick, Gelatt and Vecchi (Ref. 4).

In Section 2, we describe our method; in Section 3, we consider the numerical integration problem; and, in Section 4, we present the results of numerical experiments on several test problems.

that is, it depends on the behavior of f on each point of $\mathbb{R}^n$, and the methods that follow a trajectory of a system of ordinary differential equations are local, that is they depend only on the behavior of f along the trajectory, there is no hope of building a completely satisfactory method for global optimization based on a system of ordinary differential equations. However, the situation is different if we consider a suitable stochastic perturbation of a system of ordinary differential equations as we now describe.

Let us consider the (Ito) stochastic differential equation

$$d\xi = -\nabla f(\xi)dt + \varepsilon\, dw \qquad (2)$$

where ∇f is the gradient of f and $w(t)$ is a standard n-dimensional Wiener process. When $\varepsilon = \varepsilon_0$ is a constant, Eq. (2) is known as the Smoluchowski-Kramers equation (Ref. 3). This equation is a singular limit of the Langevin's equation when the inertial terms are neglected. The Smoluchowski-Kramers equation has been widely used by solid state physicists and chemists to study physical phenomena such as atomic migration in crystals or chemical reactions. In these applications $\varepsilon_0 = \sqrt{\frac{2kT}{m}}$ where T is the absolute temperature, k the Boltzmann constant, m the reduced mass and f the potential energy, so that (2) represents diffusion across potential barriers under the stochastic forces $\varepsilon_0 dw$.

It is well known that if $\xi^{x_0}(t)$ is the solution process of (2) starting from an initial point x_0, then the probability density function of $\xi^{x_0}(t)$ approaches, as $t \to \infty$, the limit density $\lambda e^{-f(x)/\varepsilon_0^2}$ (where λ is a normalization constant). The limit density is independent of x_0 and is peaked (indicating concentration of "particles") around the global

1. Introduction

Let $\mathbb{R}^n$ be the n-dimensional real Euclidean space $x = (x_1, x_2, \ldots, x_n)^T \in \mathbb{R}^n$ and let $f: \mathbb{R}^n \to \mathbb{R}$ be a real valued function. In this paper we consider the problem of finding the global minimizers of f, that is the points $x^* \in \mathbb{R}^n$ such that:

$$f(x^*) \leq f(x), \qquad \forall x \in \mathbb{R}^n. \tag{1}$$

A new method to numerically compute the global minimizers of f by following the paths of a system of stochastic differential equations is proposed. This method is motivated by quantum mechanics.

The importance of the global optimization problem is clear. For example, the root finding problem for the system $g(x) = 0$, where $g: \mathbb{R}^n \to \mathbb{R}^n$ can be formulated as a global optimization problem by considering the function $F(x) = \| g(x) \|_2^2$, where $\| \cdot \|_2$ is the Euclidean norm in $\mathbb{R}^n$. Despite its importance and the contributions of many researchers, the situation with respect to algorithms for the global optimization problem is still unsatisfactory and there is a need for methods with a solid mathematical foundation and good numerical performance. The situation for the problem of finding the local minimizers of f is much more satisfactory and a large body of theoretical and numerical results has been established; see for example Ref. 1 and the references given therein.

Ordinary differential equations have been used in the study of the local optimization problem or of the root finding problem by several authors; for a review see Ref. 2. These methods usually approximate the local optimizers or roots by following the trajectories of suitable systems of ordinary differential equations. However, since property (1) is a global property,

Abstract. Let $\mathbb{R}^n$ be the n-dimensional real Euclidean space, $x = (x_1, x_2, \ldots, x_n)^T \in \mathbb{R}^n$ and $f:\mathbb{R}^n \to \mathbb{R}$ be a real valued function. We consider the problem of finding the global minimizers of f. A new method to numerically compute the global minimizers by following the paths of a system of stochastic differential equations is proposed. This method is motivated by quantum mechanics. Some numerical experience on a set of test problems is presented. The method compares favorably with other existing methods for global optimization.

Key Words: Global optimization, stochastic differential equations.

JOURNAL OF OPTIMIZATION THEORY AND APPLICATIONS: Vol. 46, No. , 1985

Global Optimization and Stochastic Differential Equations[1,2]

F. ALUFFI-PENTINI[3], V. PARISI[4], AND F. ZIRILLI[5]

Communicated by R.A. Tapia

---

[1]This research has been supported by the European Research Office of the US Army under Contract No. DAJA-37-81-C-0740.

[2]The third author gratefully acknowledges Prof. A. Rinooy Kan for bringing to his attention Ref. 4.

[3]Associate Professor, Dipartimento di Matematica, Università di Bari, Bari, Italy.

[4]Ricercatore, Istituto di Fisica, 2^ Università di Roma "Tor Vergata", Roma, Italy.

[5]Professor, Istituto di Matematica, Università di Salerno, Salerno, Italy.

APPENDIX A1

Global optimization and stochastic differential equations

by F. Aluffi-Pentini, V. Parisi, and F. Zirilli

(to appear in Journal of Optimization Theory and Applications, sept. 1985).

TABLE 1

		UNIVAC					VAX				
N_{SUC} =		1	2	3	4	5	1	2	3	4	5
Totals	(1)	26	32	34	35	35	32	34	34	34	35
	(2)	0	0	0	0	0	0	0	0	0	0
	(3)	11	5	3	2	2	5	3	3	3	2
	(4)	0	0	0	0	0	0	0	0	0	0

(1)= success correctly claimed

(2)= failure correctly claimed

(3)= incorrect claim

(4)= overflow

that the performance of SIGMA is very satisfactory from the point of view of dependability (only 2 incorrect claims on the "large" dynamic range machine when $N_{SUC} > 3$ and on the "small" dynamic range machine when $N_{SUC} > 4$) and robustness (no overflows on both machines).

Unfortunately, given the state of the art of mathematical software for global optimization, it has not been possible to make conclusive comparisons with other packages.

Finally, we note that a smaller value of N_{SUC} gives a much cheaper method (less function evaluations) at the expense of a loss in effectiveness (greater number of failures).

7. Application to a problem in theoretical chemistry

SIGMA has been successfully applied to a problem in theoretical chemistry, namely the problem of finding spatial patterns of minimum intramolecular energy for a particular DNA fragment. The problem and the results are described in the paper (in italian) which is enclosed as Appendix 7.

8. Conclusion

A method for global optimization based on stochastic differential equations has been proposed, and its mathematical properties have been investigated.

A complete algorithm has been developed, which is based on following a number of simultaneously-evolving sample trajectories generated by a new stochastic scheme for numerically integrating a first-order stochastic differential equation.

The algorithm has been coded in a portable subset of the FORTRAN IV programming language, and the resulting software has been experimentally tested on a large set of test problems: 35 out of 37 problems were successfully solved, including some very difficult ones.

The software package has also been used for solving a problem in theoretical chemistry.

Working for the project has stimulated a number of scientific contacts, and the project results have been disseminated in six research papers for professional or academic journals, and in a number of seminars and communications to scientific meetings.

set of test problems, and a paper containing the problem set and the complete FORTRAN coding of the two subroutines has been submitted to the ACM Transaction on Mathematical Software (see Appendix A3).

A detailed description of the test problems and of the use of the FORTRAN subroutines is given in Appendix A3.

6. Numerical testing

SIGMA has been numerically tested on a number of test problems run on two computers.

The test problems are described in detail in Appendix 3.

The tests were performed on two typical machines of "large" and "small" dynamic range, that is, with 11 and 8 bits for the exponent (biased or signed) of double precision numbers, and corresponding dynamic range of about $10^{\pm 308}$ and $10^{\pm 38}$. The machines were:

- UNIVAC 1100/82 with EXEC8 operating system and FORTRAN (ASCII) computer (level 10R1) ("large" dynamic range)
- D.E.C. VAX 11/750 with VMS operating system (vers. 3.0) and FORTRAN compiler (vers. 3) ("small" dynamic range).

Operating conditions for the tests, and detailed results are reported in Appendix 4.

Table 1 reports summarized data concerning the effectiveness, dependability and robustness - in the form of total numbers of correctly claimed successes, correctly claimed failures, incorrect success or failure claims and total number of overflows - for the two machines and for different values of N_{SUC} (sect. 3).

The SIGMA package seems to perform quite well on the proposed test problems.

As it is shown in Annex 3 some of the test problems are very hard; for example, Problem 28 ($N = 10$) has a single global minimizer and a number of local minimizers of order 10^{10} in the region $|x_i| \leq 10 \quad i = 1,2,\ldots,10$.

Table 1 shows that from the point of view of the effectiveness as measured by the number of correctly claimed successes the performance of SIGMA is very satisfactory; moreover, it is remarkably machine independent (note that completely different pseudo-random numbers sequences are generated by the algorithm on the two test machines). The results of Table 1 also suggest

4. The software package SIGMA

The software package SIGMA is a set of FORTRAN subprograms, using double-precision floating-point arithmetics, which attempts to find a global minimizer of a real-valued function of N real variables, by means of the algorithm SIGMA, which is described in sect. 3 and in Annex A4.

The package consists of a principal subroutine SIGMA, a set of 34 auxiliary subroutines and functions, and an "easy-to-use" driver SIGMA1 which can be used to call SIGMA.

All the coding is written in FORTRAN IV and meets the specifications of PFORT, a portable subset of A.N.S. FORTRAN.

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 EXEC8 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.

The SIGMA package and its usage are described in full detail in Annex A5; the complete listing of the FORTRAN code is in Annex A6.

5. Test problems

Since the early phases of the project the need arose of experimentally testing the preliminary versions of the algorithm, in order to detect possible weak points or to compare the performance of alternative design choices. Experimental testing of an algorithm is usually performed by running its software implementation on a number of test problems: and therefore a collection of test problems naturally began to build up during project development, including problems specially conceived for the project needs by the present authors, and problems reported in the literature.

By the end of the project a final collection of 37 test problems was available: it was coded in the form of two FORTRAN subroutines, and was used for the final testing of the final version of the algorithm (sec. 6).

It was felt that the collection could be useful to the scientific community as a first attempt to provide a standard

crement 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 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 unsuccesful; 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 values 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}.

A detailed description of the algorithm is given in Appendix A5.

$$\lim_{\varepsilon_0 \to 0} p_\infty^{\varepsilon_0}(0,x_0,x) = \delta(x - x_-) \tag{13}$$

(ii) if $\Delta f_- = \Delta f_+$ and $\exists\ \alpha > 0$ such that $f(x) \geq \alpha(x-x_-)^2 + f_- \quad \forall x \quad x_0$ and $f(x) \geq \alpha(x-x_+) + f_+ \quad \forall x \geq x_0$ then

$$\lim_{\varepsilon_0 \to 0} p_\infty^{\varepsilon_0}(0,x_0,x) = \gamma\,\delta(x-x_-) + (1-\gamma)\delta(x-x_+) \tag{14}$$

where $\gamma = (1 + \sqrt{(c_-/c_+)})^{-1}$

where the limits (13), (14) are taken in the distribution sense. Proposition 2.1 is easy to prove using the Taylor formula for f around x_-, x_+.

Remark 2.1. Proposition 2.1 shows that as $\varepsilon_0 \to 0$, the asymptotic probability density approaches a Dirac delta function concentrated on the global minimizer when there is a unique global minimizer ($\Delta f_- > \Delta f_+$), or approaches a linear combination of Dirac delta functions concentrated on the global minimizers ($\Delta f_- = \Delta f_+$). The coefficients of the linear combination depend on the curvature of f at the global minimizers. These statements have a clear meaning in terms of $\xi_\infty^{\varepsilon_0}$. Finally, Proposition 2.1 can be easily generalized to a wider class of functions f.

Proposition 2.2. Under the previous hypotheses for f, Matkowsky and Schuss studied, Ref. 5., the rate of convergence of p^{ε_0} to $p_\infty^{\varepsilon_0}$ as $t \to \infty$ by looking at the eigenvalues of the Fokker-Planck operator

$$L_{\varepsilon_0}(\cdot) = \frac{\varepsilon_0^2}{2}\,\frac{\partial^2(\cdot)}{\partial x^2} + \frac{\partial}{\partial x}\left(\frac{df}{dx}\,\cdot\,\right)$$

We note that $p_\infty^{\varepsilon_0}$ is an eigenfunction with eigenvalue zero of L_{ε_0}, so that the rate of approach to $p_\infty^{\varepsilon_0}$ is determined by the next eigenvalue $\lambda_1(\varepsilon_0)$ of L_{ε_0}. Matkowsky and Shuss obtained for $\lambda_1(\varepsilon_0)$ the following asymptotic expression as $\varepsilon_0 \to 0$:

$$\lambda_1(\varepsilon_0) \simeq -\frac{\sqrt{c_+ c_0}}{2\pi}\, e^{-\frac{2}{\varepsilon_0^2}\Delta f_+} \tag{15}$$

So that roughly speaking we can imagine:

$$p^{\varepsilon_0}(0,x,t,x) = p_\infty^{\varepsilon_0} + \exp\left\{ \int_0^t \lambda_1(\varepsilon_0)ds \right\} \tilde{p} \tag{16}$$

where $\tilde{p}$ is an eigenfunction corresponding to λ_1.

When $f(x)$ is a fourth order polynomial with two minimizers, a complete analysis of the spectrum of L_{ε_0} in the limit $\varepsilon_0 \to 0$ has been given by Angeletti, Castagnari, Zirilli in Ref. 6.

Remark 2.2. Since $\lambda_1(\varepsilon_0) \to 0$ as $\varepsilon_0 \to 0$ from (16) we see that the rate of approach to $p_\infty^{\varepsilon_0}$ became slower when ε_0 became smaller. On the other hand from (12) we see that $p_\infty^{\varepsilon_0}$ becomes more and more concentrated around the global optimizers as ε_0 goes to zero.

Let us go back now to (3), (4) when $\varepsilon = \varepsilon(t)$ is a given function of t and let $\xi(t)$ be the solution of (3), (4). Let $P(0,x_0,t,A)$ be the transition probability of $\xi(t)$ and $p(0,x_0,t,x)$ the corresponding probability density. Under regularity assumptions for f, the probability density p satisfies the following Fokker-Planck equation:

$$\frac{\partial p}{\partial t} = \frac{\varepsilon^2(t)}{2} \Delta p + \operatorname{div}(\nabla f\, p) \tag{17}$$

$$\lim_{t \to 0} p(0,x_0,t,x) = \delta(x - x_0) \tag{18}$$

In order to compute the global optimizers of f by following the paths of (3), (4) we would like to show that

$$\lim_{t \to \infty} p(0,x_0,t,x) = \sum_{i=1}^{m} \gamma_i \delta(x - x_i^*) \tag{19}$$

where γ_i are positive constants such that $\sum_{i=1}^{m} \gamma_i = 1$ and x_i^*,

$i = 1,2,3, \ldots, m$ are the global minimizers of f.

The previous analysis of the correpsonding problem with $\varepsilon(t) = \varepsilon_0$ suggests that in order to have (19) we need

$$\lim_{t\to\infty} \varepsilon(t) = 0 \tag{20}$$

and, as suggested by (16), we must require that

$$\int_0^\infty e^{-\frac{2}{\varepsilon^2(t)}\Delta f_+} dt = \infty \tag{21}$$

where Δf_+ is the highest barrier to the global minimizers. We note that in order to satisfy (21) $\varepsilon(t)$ must go to zero very slowly.

The problem of giving a mathematically rigorous foundation to our method by proving (19) will be considered elsewhere. Based on the heuristic conditions (20), (21) we will consider now the problem of how to integrate numerically (3), (4) in order to obtain a global minimizer of f.

3. Numerical Integration of (3), (4)

In the previous sections we have proposed to obtain the global minimizers of f by following the paths defined by (3), (4) under suitable assumptions for $\varepsilon(t)$ when $t \to \infty$. We want to consider here the problem of how to compute numerically these paths keeping in mind that we are not really interested in the paths, but only in their asymptotic values.

The algorithm we propose here is only preliminary and further study is needed; however, as we will see in Section 4 even the present algorithm gives good numerical results on several test problems.

Let $\Delta t_k > 0$, $t_k = \sum_{i=0}^{k-1} \Delta t_i$ $(t_0 = 0)$, $k = 0,1, \ldots, ;$ we discretize (3), (4) using the Euler-Cauchy method, that is $\xi(t_k)$ is approximated by ξ_k solution of the following finite difference equations:

$$\xi_{k+1} - \xi_k = -\Delta t_k \nabla f(\xi_k) + \varepsilon(t_k)(w_{k+1} - w_k) \qquad (22)$$

$$k = 0,1, \ldots$$

$$\xi_0 = x_0. \qquad (23)$$

Since for stability reasons Δt_k will be chosen rather small and since condition (21) implies that $\varepsilon(t)$ should go to zero very slowly in order to reach the asymptotic values of the paths of (3), (4) we expect that a large number of time integration steps (22) will be needed.

Let r be an n-dimensional random vector of length 1 uniformly distributed on the $(n-1)$-dimensional sphere; then for any given non-random vector $v \in \mathbb{R}^n$, its projection $\langle v,r \rangle r$ along r is such that

$$n \cdot E(\langle v,r \rangle r) = v$$

where $E(\cdot)$ is the expected value and $<\cdot,\cdot>$ is the Euclidean inner product in $\mathbb{R}^n$. This suggests that in order to save numerical work (i.e. functions evaluations) we may substitute to $\nabla f(\xi_k)$ in eq. (22) the expression

$$n < \nabla f(\xi_k), \quad r > r \tag{24}$$

where $n < \nabla f(\xi_k), r > r$, the directional derivative in the direction r, may be further approximated by finite differences with some mesh size Δx_k.

When forward differences are used $n+1$ function evaluations are needed to approximate ∇f while only 2 function evaluations are needed to approximate the directional derivative. Finally, some heuristic algorithms are used to choose Δt_k and Δx_k to avoid instabilities. Condition (21) suggests that $\varepsilon(t)$ should go to zero very slowly as t goes to infinity so that computing a single path of (3), (4), choosing $\varepsilon(t)$ as required by (21) and following this path for a long enough period of time to obtain a global minimizer does not seem very efficient.

We have considered this alternative strategy:

(i) N paths of (3), (4) are computed ($N > 1$; $N = 7$ in the numerical experience shown in section 4) with the algorithm described before, and $\varepsilon(t)$ is kept constant.

(ii) f is computed along the paths and used as a merit functions. After a number of steps of numerical integration the N computed paths are compared. The "worst" path is discarded, the numerical integration is continued after splitting one of the remaining N-1 paths into two paths.

The new path has a different value of $\varepsilon(t) = \text{constant}$; $\varepsilon(t)$ is usually decreased, occasionally it can be increased if the paths

are stuck in a local minimizer as detected by looking at the previously computed values of f.

(iii) repeat step (ii).

4. Test Problems and Numerical Experience

The algorithm described in section 2 and 3 has been tested on a set of test problems. The first eighteen test problems have been taken from the literature, and were proposed as a set of problems to test global optimization methods by Levy and Montalvo, Ref. 7.

We shall make use of the penalization function

$$u(x,a,k,m) = \begin{cases} k(x-a)^m, & x > a, \\ 0, & -a \leq x \leq a, \\ k(x+a)^m, & x < -a. \end{cases}$$

The test problems are:

Problem 1. Goldstein's Function. Let $f(x) = x^6 - 15x^4 + 27x^2 + 250$; the function f has three minima:

$x = -3$, $f(x) = 7$,

$x = 0$, $f(x) = 250$,

$x = 3$, $f(x) = 7$.

The minimizer $x = \pm 3$ are the global minimizers of f.

Problem 2. Penalized Shubert Function. Let $g_1(x) = \sum_{i=1}^{5} i \cos((i+1)x+1)$; the function g_1 is the Shubert function. We define the penalized Shubert function $f(x)$ as follows:

$$f(x) = g_1(x) + u(x,10,100,2).$$

This function has 19 minima in the region $\{x \mid |x| < 10\}$ and three of them are global ones and they are located at:

$$x = -7.70831, \quad x = -1.42512, \quad x = 4.85805.$$

Problem 3. Two-dimensional Penalized Shubert Function. Let

$$f(x,y) = \left\{ \sum_{i=1}^{5} i \cos[(i+1)x+1] \right\} \left\{ \sum_{i=1}^{5} i \cos[(i+1)y+1] \right\}$$

$$+ u(x,10,100,2) + u(y,10,100,2)$$

The function f has 760 minima, (18 of them are global minima) in the region $\{(x,y) : |x| < 10, |y| < 10\}$.

Problem 4. Two-dimensional Penalized Shubert Function $\beta = 0.5$.

$$f(x,y) = \left\{ \sum_{i=1}^{5} i \cos[(i+1)x+1] \right\} \left\{ \sum_{i=1}^{5} i \cos[(i+1)y+1] \right\}$$

$$+ \beta\{(x+1.42513)^2 + (y+0.80032)^2\}$$

$$+ u(x,10,100,2) + u(y,10,100,2)$$

where $\beta = 0.5$ and $(-1.42513, -0.80032)$ is a point where the function f with $\beta = 0$ has a global minimizer.

This function has roughly the same behaviour of the function considered in problem 3 but has a unique global minimizer at $(-1.42513, -0.80032)$ where the function f is equal to -186.7309.

Problem 5. Two-dimensional Penalized Shubert Function $\beta = 1$. The function f is the one given in problem 4 with $\beta = 1$.

Problem 6. Camel Function. Let f be given by

$$f(x,y) = (4 - 2.1x^2 + \frac{x^4}{3})x^2 + xy + (-4 + 4y^2)y^2$$

This function has 6 minima, two of them are global minima and are located at $(-0.0898, 0.7126)$, $(0.0898, -0.7126)$.

Problems 7-9 are obtained from the following formula:

$$g_2(x) = \frac{\pi}{n}\{k_2\sin^2\pi\, y_1 + \sum_{i=1}^{n-1}[(y_i-A_2)^2(1+k_2\sin^2\pi\, y_{i+1}) + (y_n-A_2)^2]\} \tag{25}$$

where $x = (x_1,x_2, \ldots , x_n)^T$, $y_i = 1 + (x_i-1)/4$ $i = 1,2, \ldots , n$, $k_2 = 10$ $A_2 = 1$.

In the region $\Omega = \{x \in R^n \mid -10 \leq x_i \leq 10 \quad i = 1,2, \ldots , n\}$ the function (25) has roughly 5^n local minimizers and a unique global minimizer located at

$$x_i = 1 \qquad i = 1,2, \ldots , n.$$

We penalize the function (25) as follows:

$$f(x) = g_2(x) + \sum_{i=1}^{n} u(x_i,10,100,4) \tag{26}$$

Problem 7. The function $f(x)$ is given by (26) with $n = 2$.

Problem 8. The function $f(x)$ is given by (26) with $n = 3$.

Problem 9. The function $f(x)$ is given by (26) with $n = 4$.

Problems 10-12 are obtained from the following formula:

$$g_3(x) = \frac{\pi}{n}\{k_3\sin^2\pi x_1 + \sum_{i=1}^{n-1}(x_i-A_3)^2(1+k_3\sin^2\pi x_{i+1}) + (x_n-A_3)^2\} \tag{27}$$

where $k_3 = 10$, $A_3 = 1$ and $x = (x_1,x_2, \ldots , x_n)^T$.

In the region $\Omega = \{x \in \mathbb{R}^n \mid -10 \leq x_i \leq 10 \quad i = 1,2, \ldots , n\}$ the function (27) has roughly 10^n local minimizers and a unique global minimizer at $x_i = 1$ $i = 1,2, \ldots , n$. We penalize the function (27) as follows:

$$f(x) = g_9(x) + \sum_{i=1}^{n} u(x_i,10,100,4) \tag{28}$$

Problem 10. The function $f(x)$ is given by (28) with $n = 5$.

Problem 11. The function $f(x)$ is given by (28) with $n = 8$.

Problem 12. The function $f(x)$ is given by (28) with $n = 10$.

Problems 13-15 are obtained from the following formula:

$$g_4(x) = k_1\{\sin^2\pi\ell_1 x_1 + \sum_{i=1}^{n-1} (x_i - A_4)^2(1+k_2\sin^2\pi\ell_2 x_{i+1}) + (x_n - A_4)^2(1+k_2\sin^2\pi\ell_1 x_n)\} \tag{29}$$

where $k_1 = .1$, $k_2 = 1$, $A_4 = 1$, $\ell_2 = 3$, $\ell_1 = 2$.

In the region $S = \{x \in \mathbb{R}^n \mid -10 \leq x_i \leq 10 \quad i = 1,2, \ldots, n\}$ the function (29) has roughly 50^n local minimizers and a unique global minimizer at $x_i = 1$, $i = 1,2, \ldots, n$.

In the region $S_1 = \{x \in \mathbb{R}^n \mid -5 \leq x_i \leq 5 \quad i = 1,2, \ldots, n\}$ the function (29) has roughly 15^n local minimizers and a unique global minimizer at $x_i = 1 \quad i = 1,2, \ldots, n$. We penalize the function (29) as follows:

$$f(x) = g_4(x) + \sum_{i=1}^{n} u(x_i,10,100,4) \tag{30}$$

or

$$f(x) = g_4(x) + \sum_{i=1}^{n} u(x_i,5,100,4) \tag{31}$$

Problem 13. The function $f(x)$ is given by (30) with $n = 2$.

Problem 14. The function $f(x)$ is given by (30) with $n = 3$.

Problem 15. The function $f(x)$ is given by (30) with $n = 4$.

Problem 16. The function $f(x)$ is given by (31) with $n = 5$.

Problem 17. The function $f(x)$ is given by (31) with $n = 6$.

Problem 18. The function $f(x)$ is given by (31) with $n = 7$.

The problems 19-22 have been created by the third author.

Problem 19. Let $f(x) = \frac{x^4}{4} - \frac{x^2}{2} + 0.1\,x$, the function f has two minima - one for positive x and one for negative x. The one for negative x is the global one.

Problem 20. Let $f(x,y) = \frac{x^4}{4} - \frac{x^2}{2} + 0.1\,x + \frac{y^2}{2}$, the function f has two minima $(x_1,0)$, $(x_2,0)$ where x_1, x_2 are the minimizers of the function of Problem 19. The minimizer with the negative x corresponds to the global minimizer.

Problem 21. Let $f(x,y) = 0.5\,x^2 + .5\,(1 - \cos 2x) + y^2$, the function f has several local minima and the global minimizer is the origin.

Problem 22. Let $n > 0$ and $f(x,y) = 10^n x^2 + y^2 - (x^2+y^2)^2 + 10^m (x^2+y^2)^4$, the function f has a local minimum at the origin and two global minimizers on the y axis.

Problem 23. Let $f(x) = \left(\sum_{i=1}^{5} i x_i^2 \right)^{\frac{1}{4}}$ where $x = (x_1, \ldots, x_5)^T$, the function $f(x)$ has a unique minimizer at $x = 0$ where the function is not differentiable, moreover the hessian of $f(x)$ is not defined at $x = 0$ and is not positive definite in a neighborhood of $x = 0$.

The remaining problem 24 has been suggested by S. Wolff, Ref. 8.

Problem 24. Let

$$F(x,y) = -F(x,y) + u(x,10^{4},100,2) + u(y,10^{4},100,2)$$

where $F(x,y) = \prod_{i=1}^{24} [\Phi(\frac{x_i - x}{y})]^{1-\delta_i} [1 - \Phi(\frac{x_i - x}{y})]^{\delta_i}$

the data points x_i δ_i are given by:

x_i	1219	1371	1377	1144	1201	1225	1244
δ_i	0	0	0	1	1	1	1
x_i	1294	1304	1328	1351	1356	1370	1390
δ_i	1	1	1	1	1	1	1

and $\Phi(x) = \int_{-\infty}^{x} e^{-t^2/2} \frac{dt}{\sqrt{2\pi}}$.

The function $f(x,y)$ has an absolute minimizer at (1523.2, 277.5) and a spurious relative minimizer due to the penalization at $(-6607.3, -10^{4})$.

The numerical results obtained are shown in Table 1.

TABLE 1

Problem	NF1	Whether a global minimizer has been found	NF2	Whether a global minimizer has been found	Remarks
1	3,184	Yes	7,168	Yes	
2	26,893	Yes	77,699	Yes	
3	3,218	No	241,215	Yes	
4	8,755	Yes	76,894	Yes	
5	97,761	Yes	183,819	Yes	
6	5,393	Yes	10,822	Yes	
7	84,782	Yes	159,549	Yes	
8	19,041	Yes	72,851	Yes	
9	18,942	Yes	49,690	Yes	
10	18,433	Yes	72,226	Yes	
11	4,322	No	136,061	Yes	
12	49,701	Yes	98,985	Yes	
13	9,492	Yes	23,770	Yes	
14	19,114	Yes	66,010	Yes	
15	35,139	Yes	122,166	Yes	
16	53,398	Yes	66,365	Yes	
17	15,534	Yes	98,974	Yes	
18	16,542	Yes	109,886	Yes	
19	6,751	Yes	16,487	Yes	
20	3,402	Yes	12,249	Yes	
21	10,286	Yes	19,940	Yes	
22	4,791	Yes	7,390	Yes	n=-m=1
22	3,037	Yes	4,853	Yes	n=-m=2
22	5,028	Yes	8,235	Yes	n=-m=3
22	14,710	Yes	27,859	Yes	n=-m=4
22	51,285	Yes	74,194	Yes	n=-m=5

22	17,610	Yes	4,042,861	Yes	n=-m=6
23	15,102	Yes	34,110	Yes	
24	48,802	Yes	69,512	Yes	

The program is run twice on each problem, the first time with a given stopping criterion. NF1 is the number of function evaluations (including the ones needed to evaluate the gradient) used in this first run while the result obtained is shown in column 3. The second time the program is run with a more stringent stopping criterion and the columns 4, 5, have the same meaning as columns 2, 3, respectively. All the remaining parameters (initial value for (t) etc. ...) are fixed once and for all during the runs.

The initial point x_0 has been chosen as follows:

$x_0 = 0$ for Problems 1-18

$x_0 = 1/50$ for Problem 19

$x_0 = (1,0)$ for Problem 20

$x_0 = (-3,0)$ for Problem 21

$x_0 = (0,1)$ for Problem 22

$x_0 = (10^2, 10^2, \ldots, 10^2)$ for Problem 23

$x_0 = (-1250, -1000)$ for Problem 24.

For Problems 19-22 and 24 the initial point x_0 has been chosen close to a local minimizer.

The condition number of the Hessian at the solution of Problem 22 increases with n,-m; the Hessian at the solution of Problem 23 is not defined; the Hessian at the solution of Problem 24 is ill-conditioned; the remaining problems have well-conditioned Hessians at the solutions.

$$y = \pm \frac{1}{\varepsilon} \left(\frac{4\alpha^2 - \sqrt{4\alpha^4+9\varepsilon^2}}{3} \right)^{1/2}$$

(ii) since $V_\varepsilon(y)$ is even let us consider only $y > 0$, by explicit computation it is easy to obtain the following table:

y	0	$\sqrt{\frac{4\alpha^2-\sqrt{4\alpha^4+9\varepsilon^2}}{3\varepsilon^2}}$	$\frac{\alpha\sqrt{6}}{3\varepsilon}$	$\frac{\alpha\sqrt{2}}{\varepsilon}$	$\sqrt{\frac{4\alpha^2+\sqrt{4\alpha^4+9\varepsilon^2}}{3\varepsilon^2}}$
$V_\varepsilon(y)$	$2\varepsilon^2$	$\frac{2}{27\varepsilon^2}\left(8\alpha^6-27\alpha^2\varepsilon^2+\right.$ $\left.+(4\alpha^4+9\varepsilon^2)^{3/2}\right)$	$\frac{32\alpha^6}{27\varepsilon^2}$	$-4\alpha^2$	$\frac{2}{27\varepsilon^2}\left(8\alpha^6-27\alpha^2\varepsilon^2-\right.$ $\left.-(4\alpha^4+9\varepsilon^2)^{3/2}\right)$
$V'_\varepsilon(y)$	0	0	$-2\alpha\varepsilon\sqrt{6}$	$-6\alpha\varepsilon\sqrt{2}$	0
$V''_\varepsilon(y)$	$8\alpha^4 - 6\varepsilon^2$	$\frac{8}{3}\left((4\alpha^4+9\varepsilon^2) - \right.$ $\left.-4\alpha^2\sqrt{4\alpha^4+9\varepsilon^2}\right)$	$-\frac{32}{3}\alpha^4 - 6\varepsilon^2$	$32\alpha^4-6\varepsilon^2$	$\frac{8}{3}\left((4\alpha^4+9\varepsilon^2) + \right.$ $\left.+ 4\alpha^2\sqrt{4\alpha^4+9\varepsilon^2}\right)$

Table 1

(where $\cdot'$ means differentiation)

(iii) $V_\varepsilon(y)$ is bounded below by a constant independent of ε

(iv) $V_\varepsilon(y)$ is given by Fig. 3.

$$(2.20)\qquad U_\varepsilon(y) = \tfrac{1}{2}\left(\tfrac{1}{2}\left(\frac{df_{2\varepsilon}}{dy}\right)^2 - \frac{d^2f_{2\varepsilon}}{dy^2}\right) =$$

$$= \tfrac{1}{4}y^2(2a\varepsilon^2y^2 + \frac{3b\varepsilon}{\sqrt{2}}y + 2c)^2 - \tfrac{1}{2}(6a\varepsilon^2y^2 + \frac{6b\varepsilon}{\sqrt{2}}y + 2c) =$$

$$= a^2\varepsilon^4y^6 + \frac{3ab}{\sqrt{2}}\varepsilon^3y^5 + (\tfrac{9}{8}b^2+2ac)\varepsilon^2y^4 + \frac{3bc}{\sqrt{2}}\varepsilon y^3 +$$

$$+ (c^2-3a\varepsilon^2)y^2 - \frac{3b\varepsilon}{\sqrt{2}}y - c$$

In order to understand intuitively the behavior as $\varepsilon \to 0$ of the spectrum of H_ε when the potential $W_\varepsilon(y)$ is given by V_ε or U_ε let us analyze the behavior of V_ε and U_ε when $\varepsilon \to 0$.

Proposition 2.1. Let $V_\varepsilon(y)$ be given by (2.19), then $V_\varepsilon(y)$ is an even sixth degree polynomial. There exists $\varepsilon_0 > 0$ such that for $0 < \varepsilon < \varepsilon_0$:

(i) the equation

$$(2.21)\qquad \frac{dV_\varepsilon}{dy}(y) = 0$$

has five real roots

$$y = 0 \qquad y = \pm\frac{1}{\varepsilon}\left(\frac{4\alpha^2 \pm \sqrt{4\alpha^4+9\varepsilon^2}}{3}\right)^{1/2}$$

that is, V_ε has a local minimizer at $y = 0$, two global minimizers at $y = \pm\frac{1}{\varepsilon}\left(\frac{4\alpha^2 + \sqrt{4\alpha^4+9\varepsilon^2}}{3}\right)^{1/2}$ and two local maximizers at

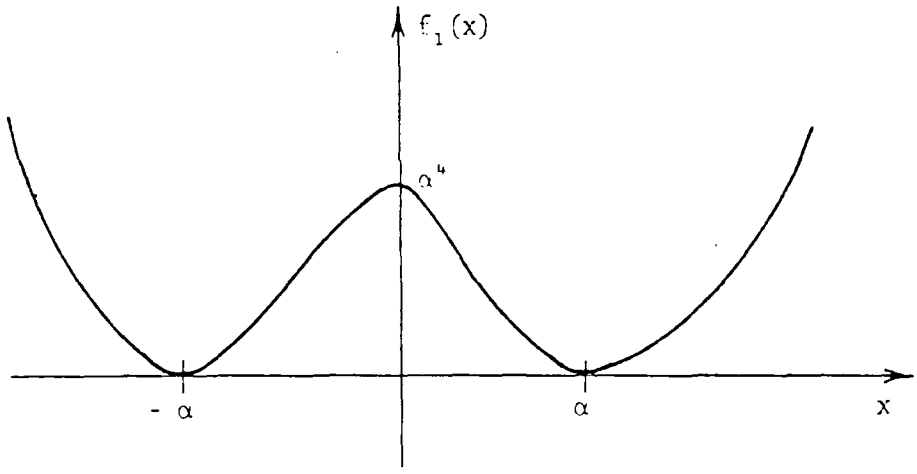


Fig. 1

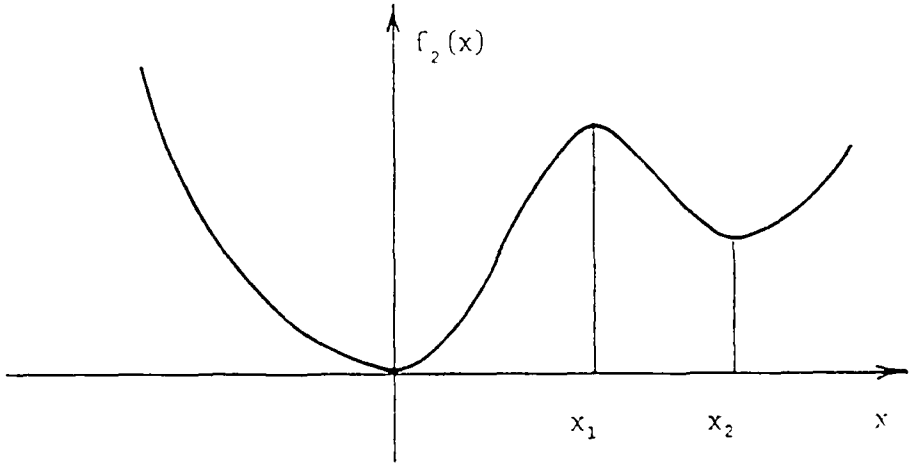


Fig. 2

In this paper we will consider the case when $f(x)$ is given by

$$f_1(x) = (x^2-a^2)^2 \qquad a > 0, \quad x \in \mathbb{R} \tag{2.13}$$

or by

$$f_2(x) = x^2(ax^2+bx+c) \qquad x \in \mathbb{R} \tag{2.14}$$

where $a > 0$, a, b, c, are real constants and

$$a > 0 \tag{2.15}$$

$$b^2-4ac < 0 \tag{2.16}$$

$$9b^2-32ac > 0 \tag{2.17}$$

$$b < 0 \tag{2.18}$$

Since the spectrum of H_ε is invariant with respect to adding a constant to f, to making translation on the x-axis, or to changing x into $-x$, $f_1(x)$ represents the most general fourth degree polynomial with two global minimizers (Fig. 1) and $f_2(x)$ represents the most general fourth degree polynomial with one global minimizer and one local minimizer. Let us remark that (2.15), (2.16) implies that $f_2(x) \geq 0 \quad \forall x \in \mathbb{R}$, with $f_2(x) = 0 \iff x = 0$, (2.17) implies that $f_2'(x) = 0$ has three real roots $0, x_1, x_2$ and that $f_2''(x) = 0$ has two real roots, that is x_1 is a maximizer of f_2 and x_2 is a minimizer of $f_2(x)$; finally (2.18) implies that $0 < x_1 < x_2$ (Fig. 2).

A straightforward computation gives:

$$V_\varepsilon(y) = \frac{1}{4\varepsilon^2}\left\{\left(\frac{df_\varepsilon}{dy}\right)^2 - \varepsilon\frac{d^2f_\varepsilon}{dy^2}\right\} =$$

$$= \varepsilon^2 y^6 - 4a\varepsilon^2 y^4 + (4a^2-3b\varepsilon)y^2 + 2a\varepsilon$$

Let us note that H_ε is a Schrodinger hamiltonian. It is easy to verify that

$$v_0(y) = c_\varepsilon^{\frac{1}{2}} e^{-f_\varepsilon(y)/2} \qquad y \in \mathbb{R} \tag{2.8}$$

is a solution of (2.5) when $\lambda = 0$, corresponding to $v_0(y)$ we have

$$u_0(x) = c_\varepsilon e^{-2/\varepsilon^2 f(x)} \qquad x \in \mathbb{R} \tag{2.9}$$

solution of (2.1) when $\lambda = 0$. Since we would like to interpret $u_0(x)$ as the probability density of a random variable we will assume that

$$\int_{-\infty}^{+\infty} e^{-2/\varepsilon^2 f(x)} dx < \infty \qquad \forall \varepsilon \neq 0 \tag{2.10}$$

and we will choose

$$c_\varepsilon = \left(\int_{-\infty}^{+\infty} e^{-2/\varepsilon^2 f(x)} dx \right)^{-1} \tag{2.11}$$

so that

$$\int_{-\infty}^{+\infty} u_0(x) dx = 1 \tag{2.12}$$

Condition (2.12) means that $u_0(x) \in L^1(\mathbb{R})$ this implies that $v_0(y) \in L^2(\mathbb{R})$ where $L^p(\mathbb{R})$ is the Lebesgue space of index p, so that it is natural to study the spectrum of H_ε in $L^2(\mathbb{R})$.

§2. From the Fokker-Planck equation to the Schrodinger equation.

Let us consider the eigenvalue problem

$$L_\varepsilon(u) = \lambda u \qquad \lambda \in \mathbb{C}, \quad x \in \mathbb{R} \tag{2.1}$$

where L_ε is given by (1.4).

Let us consider the change of variables

$$y = \frac{\sqrt{2}}{\varepsilon} x \tag{2.2}$$

$$v(y) = c_\varepsilon^{-1} e^{f_\varepsilon(y)/2} u\left(\frac{\varepsilon}{\sqrt{2}} y \right) \tag{2.3}$$

where c_ε is a normalization constant and

$$f_\varepsilon(y) = \frac{2}{\varepsilon^2} f\left(\frac{\varepsilon}{\sqrt{2}} y \right) \tag{2.4}$$

The eigenvalue problem (2.1) becomes

$$H_\varepsilon v = -\lambda v \qquad \lambda \in \mathbb{C}, \quad y \in \mathbb{R} \tag{2.5}$$

where

$$H_\varepsilon = -\frac{d^2}{dy^2} + W_\varepsilon(y) \tag{2.6}$$

and

$$W_\varepsilon(y) = \frac{1}{4}\left(\frac{df}{dy} \right)^2 - \frac{d^2 f}{dy^2} \tag{2.7}$$

The interest of one of us (F.Z.) in the study of the asymptotic behavior of the spectrum of the Fokker-Planck operators arose in the study of a method for global optimization based on the use of suitable stochastic differential equations [11].

In §2 the eigenvalue problem for L_ε is reduced to an eigenvalue problem for a suitable Schrodinger hamiltonian H_ε. The particular Schrodinger hamiltonian obtained when f is a fourth degree polynomial with two minimizers are studied in detail.

In §3 some approximating hamiltonians that will be used later are introduced and studied.

In §4 all the basic estimates needed to prove our main results are proved.

In §5 a theorem concerning the behavior as $\varepsilon \to 0$ of the difference between the resolvent of H_ε and the resolvent of the approximating hamiltonian is proved.

Moreover the asymptotic behavior as $\varepsilon \to 0$ of the spectrum of H_ε and as a consequence of the spectrum of L_ε is considered.

In §6 using the Rayleigh-Ritz principle for H_ε a particularly simple asymptotic formula for the first nonzero eigenvalue of L_ε is obtained.

Finally in §7 the case when f is given by a general smooth function is considered formally and some conclusions are drawn.

where $P_r\{\cdot\}$ = Probability of $\{\cdot\}$, $p_\varepsilon(x,x_0,t)$ is the solution of the Fokker-Planck equation:

$$\frac{\partial p}{\partial t} = L_\varepsilon(p) \qquad x \in \mathbb{R}, \quad t > 0 \tag{1.3}$$

where $L_\varepsilon(\cdot)$, the Fokker-Planck operator, is given by:

$$L_\varepsilon(p) = \frac{\varepsilon^2}{2} \frac{\partial^2 p}{\partial x^2} + \frac{\partial}{\partial x}\left(\frac{\partial f}{\partial x} p \right) \quad x \in \mathbb{R} \tag{1.4}$$

subject to the condition

$$\lim_{t\to 0} p_\varepsilon(x,x_0,t) = \delta(x-x_0) \tag{1.5}$$

where $\delta(\cdot)$ is the Dirac's delta.

The problem of deriving asymptotic formulas as $\varepsilon \to 0$ for t' first nonzero eigenvalue of the Fokker-Planck operator has been considered for a long time both on physical and mathematical grounds. We refer for reasons of brevity only to the recent paper by Matkowsky and Schuss [10] where several Fokker-Planck operators including some two-dimensional ones are considered.

However, the problem of studying the spectrum of the Fokker-Planck operator as $\varepsilon \to 0$ has received much less attention. In this paper we restrict our attention to the one-dimensional case when L_ε is given by (1.4) and f is a fourth degree polynomial with two minimizers.

Even in this particular case the resulting problems is an interesting singular perturbation problem for the ordinary differential operator L_ε.

§1. Introduction

Asymptotic eigenvalue degeneracy due to singular perturbations is a common phenomenon to many different fields of applied mathematics such as quantum mechanics [1], [2], [3], [4], [5]. [6], statistical mechanics and quantum field theory [7].

In this paper we study the behavior as the diffusion constant goes to zero of the spectrum of a class of one-dimensional Fokker-Planck operators. The problem considered here can be considered analogous for the Fokker-Planck equation of the anharmonic oscillator problem for the Schrodinger equation studied in [1], [2], [4]. In particular we will follow the path of Isaacson in [2].

Let us consider the Smoluchowski approximation to Langevin's equation [8], [9]:

$$dx(t) = -\nabla f(x(t))dt + \varepsilon \, dw(t) \tag{1.1}$$

where $f:\mathbb{R} \to \mathbb{R}$ is a smooth function called potential, $\mathbb{R}$ is the real line, ε is a real parameter, $w(t)$ is a standard one-dimensional Wiener process. The equation (1.1) is an Ito stochastic differential equation widely used in mathematical physics and engineering whose solution $x_\varepsilon(t)$ is a stochastic process.

The transition probability density $p_\varepsilon(x,x_0,t)$ of $x_\varepsilon(t)$ is defined as:

$$p_\varepsilon(x,x_0,t)dx \equiv P_r\{x_\varepsilon(t) \in (x,x+dx) \mid x_\varepsilon(0) = x_0\} \tag{1.2}$$

Asymptotic eigenvalue degeneracy for a class of one-dimensional Fokker-Planck operators[†]

Angelo Angeletti

Istituto Matematico, Universitá di Camerino, 62032 Camerino (MC) Italy

Cinzia Castagnari

Istituto Matematico, Universitá di Camerino, 62032 Camerino (MC) Italy

Francesco Zirilli[‡]

Department of Mathematical Sciences, Rice University, Houston, Texas 77001 USA

† The research reported in this document has been made possible through the support and sponsorhip of the U.S. Government through its European Research Office of the U.S. Army under contract n. DAJA-37-81-C-0740

‡ Permanent address: Istituto Matematico, Universitá di Salerno, 84100 Salerno (Italy)

APPENDIX A2

Asymptotic eigenvalue degeneracy for a class of one-dimensional Fokker-Planck operators

by A. Angeletti, C. Castagnari, F. Zirilli

(to appear in Journal of Mathematical Physics).

References

1. POWELL, M.J.D., Editor, Nonlinear Optimization 1981, Academic Press, New York, New York, 1982.

2. ZIRILLI, F., The Use of Ordinary Differential Equations in the Solution of Nonlinear Systems of Equations, Nonlinear Optimization 1981, Edited by M.J.D. Powell, Academic Press, New York, New York, 1982.

3. SCHUSS, Z., Theory and Applications of Stochastic Differential Equations, John Wiley and Sons, New York, New York, Chapter 8, 1980.

4. KIRKPATRICK, S., GELATT, C.D. Jr., and VECCHI, M.P., Optimization by Simulated Annealing, Science, Vol. 220, pp. 671-680, 1983.

5. MATKOWSKY, B.J., SCHUSS, Z., Eigenvalues of the Fokker-Planck Operator and the Approach to Equilibrium for Diffusions in Potential Fields, SIAM Journal of Applied Mathematics, Vol. 40, pp. 242-254, 1981.

6. ANGELETTI, A., CASTAGNARI, C., ZIRILLI, F., Asymptotic Eigenvalue Degeneracy for a Class of One-dimensional Fokker-Planck Operators, to appear in Journal of Mathematical Physics.

7. LEVY, A.V., and MONTALVO, A., Algoritmo de Tunelizacion para la Optimizacion Global de Funciones, Universidad Nacional Autonoma de México, Instituto de Investigaciones en Matematicas Aplicadas y en Sistemas, Report No. 204, 1979.

8. WOLFF, S., Private Communication.

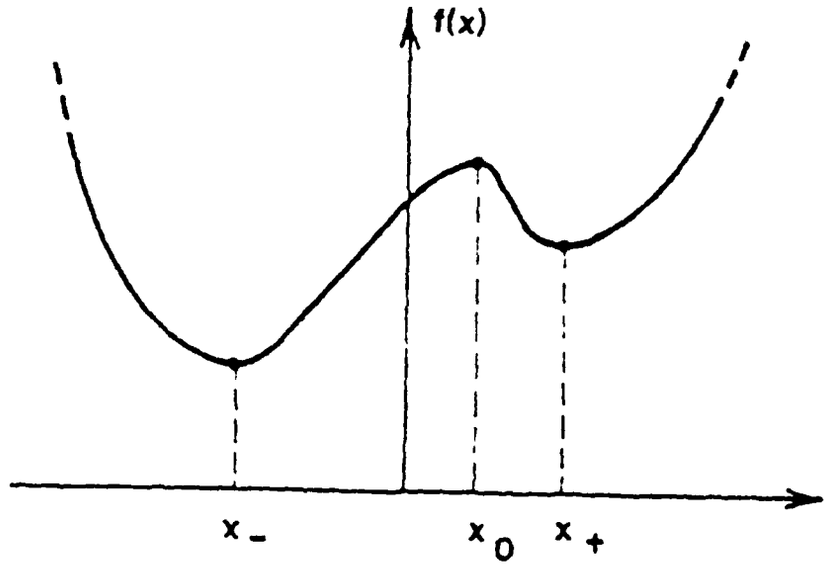


FIGURE 1

The numerical experience contained in table 1 shows that the present implementation of our method is much more sensitive to ill-conditioning than to the total number of local minimizers. This seems to be due to the method used to numerically integrate the stochastic differential equations. However, we should remark that on Problems 10, 11, 12, 16, 17, 18 that have a very large number of local minimizers the global one is obtained by using a number of function evaluations much smaller than the number of local minimizers. Our method gives satisfactory results on all the test problems including Problem 23 that is not differentiable at the solution. Finally, we note that given the stochastic nature of the method the amount of work needed to solve a problem depends on the problem and on the sequence of random numbers generated during the numerical integration.

We feel that further work both of mathematical and numerical character must be spent on the ideas presented in this paper.

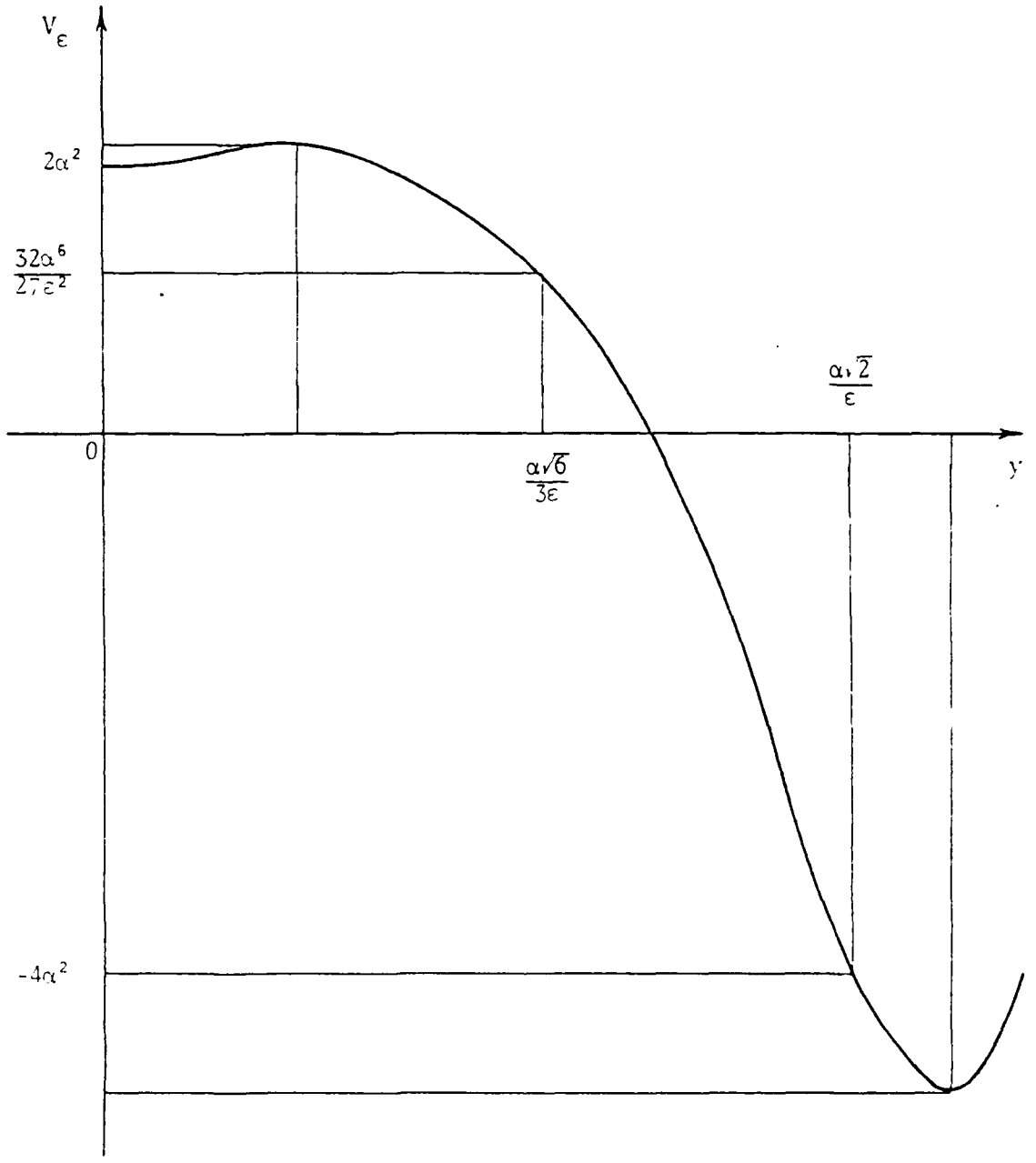

V_ε
$2\alpha^2$
$\frac{32\alpha^6}{27\varepsilon^2}$
0
$\frac{\alpha\sqrt{6}}{3\varepsilon}$
$\frac{\alpha\sqrt{2}}{\varepsilon}$
y
$-4\alpha^2$

Fig. 3

Proposition 2.2. Let $U_\varepsilon(y)$ be the sixth degree polynomial given by (2.20). There exists $\varepsilon_0 > 0$ such that for $0 < \varepsilon < \varepsilon_0$:

(i) we can consider the points

$$y = 0, \quad y_1 = \frac{\sqrt{2}}{\varepsilon} x_1, \quad y_2 = \frac{\sqrt{2}}{\varepsilon} x_2$$

where $x_{1,2} = \frac{-3b \mp \sqrt{9b^2 - 32ac}}{8a}$ are such that $\frac{df_2}{dx}(x_{1,2}) = 0$ and the points

$$\eta_1 = \frac{\sqrt{2}}{\varepsilon} \xi_1 , \quad \eta_2 = \frac{\sqrt{2}}{\varepsilon} \xi_2$$

where $\xi_{1,2} = \frac{-3b \mp \sqrt{9b^2 - 24ac}}{12a}$ are such that $\frac{d^2 f_2}{dx^2}(\xi_{1,2}) = 0$.

Let us remark that (2.15), (2.16), (2.17) imply that $\xi_{1,2}$ are real (i.e. $9b^2 - 24ac > 0$). Moreover $0 < \xi_1 < x_1 < \xi_2 < x_2$ so that $0 < \eta_1 < y_1 < \eta_2 < y_2$.

(ii) we have

$$U'_\varepsilon(y) = \tfrac{1}{2}(f'_{2\varepsilon} f''_{2\varepsilon} - f'''_{2\varepsilon}) \tag{2.22}$$

$$U''_\varepsilon(y) = \tfrac{1}{2}[(f''_{2\varepsilon})^2 + f'_{2\varepsilon} f'''_{2\varepsilon}] - \tfrac{1}{2} f^{(iv)}_{2\varepsilon} \tag{2.23}$$

where $\cdot'$ means differentiation.

(iii) by explicit computation from (ii) it is easy to obtain the following table:

y	0	η_1	y_1	η_2	y_2
$U_\varepsilon(y)$	$-c$	$\frac{1}{2c^2}(f_2'(\xi_1))^2$	$-c_1$	$\frac{1}{2c^2}(f_2'(\xi_2))^2$	$-c_2$
$U_\varepsilon'(y)$	$-\frac{3b}{\sqrt{2}}$	$-\frac{1}{2}\frac{\varepsilon}{\sqrt{2}}f_2'''(\xi_1)$	$-\frac{1}{2}\frac{\varepsilon}{\sqrt{2}}f_2'''(x_1)$	$-\frac{1}{2}\frac{\varepsilon}{\sqrt{2}}f_2'''(\xi_2)$	$-\frac{1}{2}\frac{\varepsilon}{\sqrt{2}}f'''(x_2)$
$U_\varepsilon''(y)$	$\frac{1}{2}(c^2-3ac^{\frac{4}{3}})$	$\frac{1}{2}[f_2'(\xi_1)f_2'''(\xi_1)$ $-\frac{\varepsilon^2}{2}f_2^{(iv)}(\xi_1)]$	$2c_1^2-$ $\frac{\varepsilon^2}{4}f_2^{(iv)}(x_1)$	$\frac{1}{2}[f_2'(\xi_2)f'''(\xi_2)$ $-\frac{\varepsilon^2}{2}f_2^{(iv)}(\xi_2)]$	$2c_2^2-$ $\frac{\varepsilon^2}{4}f^{(iv)}(x_2)$

Table 2

where $c_1 = \frac{1}{2}\frac{d^2f_2}{dx^2}(x_1) < 0$, $c_2 = \frac{1}{2}\frac{d^2f_2}{dx^2}(x_2) > 0$. Moreover $c = 2ax_1x_2$, $c_1 = 2ax_2(x_1-x_2)$ and $c_2 = 2ax_1(x_2-x_1)$.

(iv) from Table 2 we can deduce that the equation

$$\frac{dU_\varepsilon}{dy} = 0$$

has five real roots so that $U_\varepsilon(y)$ has three minimizers and two maximizers.

(v) $U_\varepsilon(y)$ is bounded below by a constant independent of ε.

(vi) $U_\varepsilon(y)$ is given by Fig. 4.

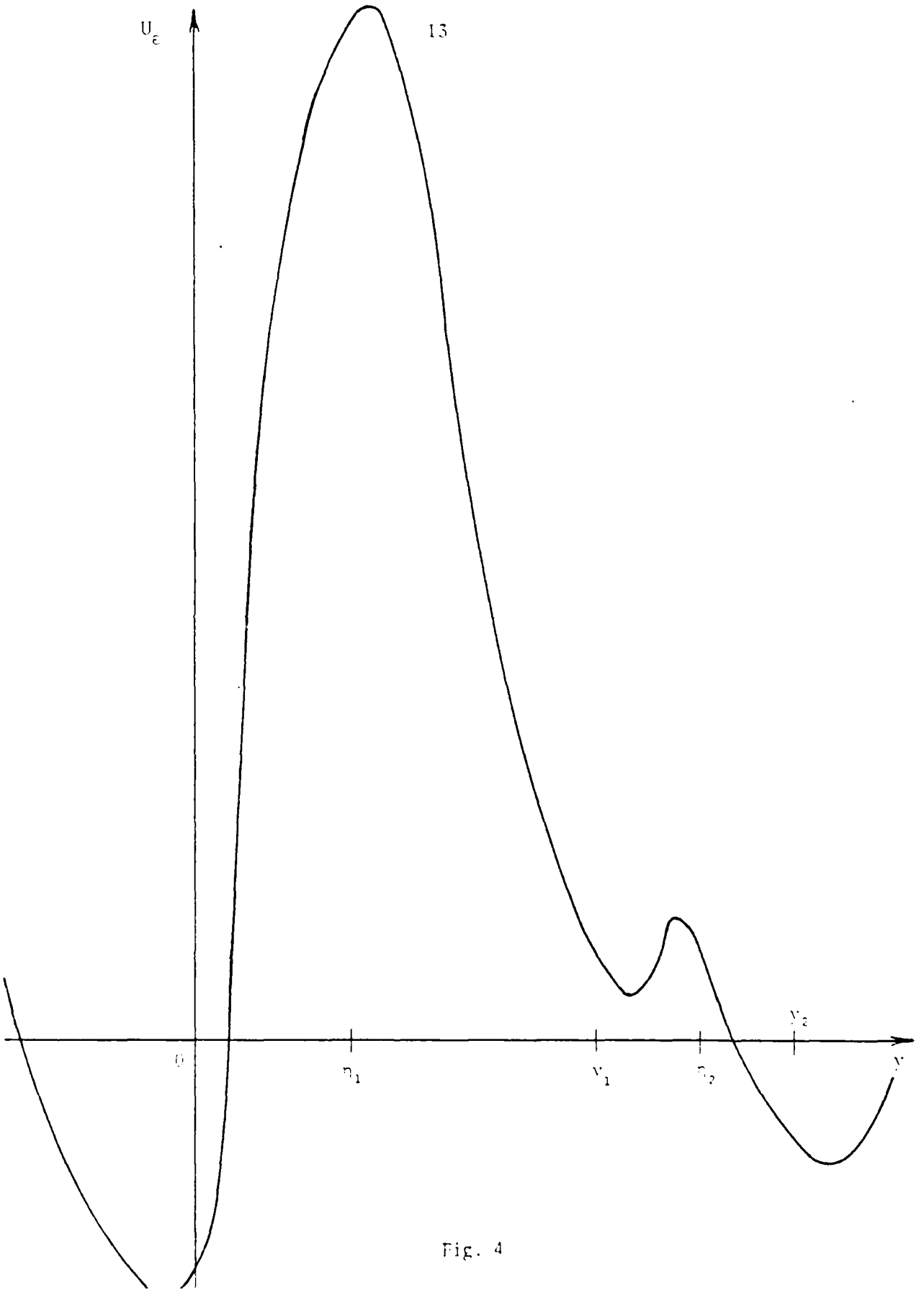


Fig. 4

From Proposition 2.1 and Fig. 3 it follows that as $\varepsilon \to 0$ $V_\varepsilon(y)$ approaches three independent harmonic oscillator potentials, one with vertex at $y = 0$ and equation $4\alpha^4 y^2 + 2\alpha^2$ and two with vertices at $y = \pm \frac{\sqrt{2}}{\varepsilon}\alpha$ and equations $16\alpha^4(y \mp \frac{\sqrt{2}}{\varepsilon}\alpha)^2 - 4\alpha^2$.

Let H_ε be given by (2.6) and $W_\varepsilon(y) = 4\alpha^4 y^2 + 2\alpha^2$ then the eigenvalues in (2.5) are given by

$$-\lambda_n^{(1)} = 4\alpha^2(n+1) \qquad n = 0,1,2, \ldots \tag{2.24}$$

the eigenvalues corresponding to the remaining two harmonic oscillators are

$$-\lambda_n^{(2)} = 8\alpha^2 n \qquad n = 0,1,2, \ldots \tag{2.25}$$

$$-\lambda_n^{(3)} = 8\alpha^2 n \qquad n = 0,1,2, \ldots \tag{2.26}$$

In section 5 we will prove that the eigenvalues of

$$M_\varepsilon = -\frac{d^2}{dy^2} + V_\varepsilon(y) \qquad y \in \mathbb{R} \tag{2.27}$$

approache (2.24), (2.25), (2.26) when $\varepsilon \to 0$. In particular we will show that the first eigenvalue $\lambda_0 = 0$ as $\varepsilon \to 0$ has asymptotically multiplicity 2 (i.e. $\lambda_1(\varepsilon) \to 0$ when $\varepsilon \to 0$) as can be seen from (2.25), (2.26) when $n = 0$. Moreover

$$\lim_{\varepsilon \to 0} -\lambda_{2+4n}(\varepsilon) = 4\alpha^2(2n+1) \qquad n = 0,1,2 \tag{2.28}$$

as can be seen from (2.24) and

$$(2.29)\qquad \lim_{\varepsilon\to 0} -\lambda_{3+4n}(\varepsilon) = \lim_{\varepsilon\to 0} -\lambda_{4+4n}(\varepsilon) = \lim_{\varepsilon\to 0} -\lambda_{5+4n}(\varepsilon) = 8\alpha^2(n+1)$$

$$n = 0,1,2, \ldots$$

as can be seen from (2.24), (2.25), (2.26). So that M_ε as $\varepsilon \to 0$ has eigenvalues with multiplicity one (i.e. the ones coming from (2.28)) and eigenvalues with asymptotic multiplicity three (i.e. the ones coming from (2.29)).

From Proposition 2.2 and Fig. 4 it follows that as $\varepsilon \to 0$ $U_\varepsilon(y)$ approaches three independent harmonic oscillator potentials one with vertex at $y = 0$ and equation $c^2y^2 - c$, one with vertex at $y = y_1$ and equation $c_1^2(y-y_1)^2 - c_1$ $(c_1 < 0)$, and one with vertex at $y = y_2$ and equation $c_2^2(y-y_2)^2 - c_2$ $(C_2 > 0)$.

Let H_ε be given by (2.6) and $W_\varepsilon(y) = c^2y^2 - c$ then the eigenvalues in (2.5) are given by

$$(2.30)\qquad -\bar{\lambda}_n^{(1)} = (2n+1)\,c - c \qquad n = 0,1,2, \ldots \qquad (c > 0)$$

the eigenvalues corresponding to the remaining two harmonic oscillators are

$$(2.31)\qquad -\bar{\lambda}_n^{(2)} = (2n+1)|c_1| - c_1 \qquad n = 0,1,2, \ldots \qquad (c_1 < 0)$$

$$(2.32)\qquad -\bar{\lambda}_n^{(3)} = (2n+1)\,c_2 - c_2 \qquad n = 0,1,2, \ldots \qquad (c_2 > 0)$$

In section 5 we will prove that the eigenvalues of

$$(2.33)\qquad N_\varepsilon = -\frac{d^2}{dy^2} + U_\varepsilon(y) \qquad y \in \mathbb{R}$$

approaches (2.30), (2.31), (2.32) when $\varepsilon \to 0$. In particular we will show that the first eigenvalue $\lambda_1 = 0$ as $\varepsilon \to 0$ has asymptotically multiplicity 2 (i.e. $\lambda_2(\varepsilon) \to 0$ when $\varepsilon \to 0$). The remaining eigenvalues, since c, c_1, c_2 can be expressed in terms of a, x_1, x_2 as shown in Proposition 2.2 (iii), have multiplicity one if $\frac{x_1}{x_2}$ is irrational, have multiplicity one or three if $\frac{x_1}{x_2}$ is rational.

§3. The approximating hamiltonians.

Let $C_0^\infty(\mathbb{R})$ be the space of the infinitely differentiable functions of compact support. Let $h_0 : \mathcal{D}(h_0) \subset L^2(\mathbb{R}) \to L^2(\mathbb{R})$ denote the self-adjoint extension of $-\frac{\partial^2}{\partial y^2}$ and let $\mathcal{D}(y^m)$ denote the domain of the self-adjoint multiplication operator y^m.

The Schrodinger hamiltonians M_ε, N_ε as operators on $L^2(\mathbb{R})$ possess the following properties:

Theorem 3.1. For any $\varepsilon \in \mathbb{R}$ with $\varepsilon \neq 0$

(i) M_ε is essentially self-adjoint on $C_0^\infty(\mathbb{R})$ and is self-adjoint on $\mathcal{D}(h_0) \cap \mathcal{D}(y^6)$

(ii) M_ε has compact resolvent

(iii) the eigenvalues of M_ε are non degenerate

(iv) the eigenfunctions alternate parity and the one corresponding to the smallest eigenvalue is even.

Proof: See [6] and [12].

Theorem 3.2: For any $\varepsilon \in \mathbb{R}$ with $\varepsilon \neq 0$

(i) N_ε is essentially self-adjoint on $C_0^\infty(\mathbb{R})$ and is self-adjoint on $\mathcal{D}(h_0) \cap \mathcal{D}(y^6)$

(ii) N_ε has compact resolvent

(iii) the eigenvalues of N_ε are non degenerate.

Proof: See [6] and [12].

Let $A_+ = \{y | y > \frac{\alpha\sqrt{6}}{3\varepsilon}\}$, $A_0 = \{y \big| |y| < \frac{\alpha\sqrt{6}}{3\varepsilon}\}$, $A_- = \{y | y < -\frac{\alpha\sqrt{6}}{3\varepsilon}\}$ and define $V_{2\varepsilon}$ as follows:

$$V_{2\varepsilon}(y) = \begin{cases} 4\alpha^4 \left(y - \frac{\alpha\sqrt{6}}{3\varepsilon} - \frac{1}{2\nu} \frac{1}{y - \frac{\alpha\sqrt{6}}{3\varepsilon}} \right)^2 - 4\alpha^2 & \text{when } y \in A_+ \\ \frac{V_0}{\cos^2 \beta y} - V_0 + 2\alpha^2 & \text{when } y \in A_0 \\ 4\alpha^4 \left(y + \frac{\alpha\sqrt{6}}{3\varepsilon} - \frac{1}{2\nu} \frac{1}{y + \frac{\alpha\sqrt{6}}{3\varepsilon}} \right)^2 - 4\alpha^2 & \text{when } y \in A_- \end{cases} \tag{3.1}$$

(see Fig. 5)

where

$$\frac{1}{2\nu} = \left(\frac{\alpha\sqrt{2}}{\varepsilon} - \frac{\alpha\sqrt{6}}{3\varepsilon} \right)^2 = \frac{2\alpha^2}{\varepsilon^2} \left(\frac{3 - \sqrt{3}}{3} \right)^2 \tag{3.2}$$

$$\beta = \frac{\pi}{2} \frac{3\varepsilon}{\alpha\sqrt{6}} \tag{3.3}$$

$$V_0 = \frac{32}{3} \frac{\alpha^6}{\pi^2 \varepsilon^2} \tag{3.4}$$

The function $V_{2\varepsilon}$ as $\varepsilon \to 0$ is an approximation to V_ε in particular $V_{2\varepsilon}$ approaches three independent harmonic oscillator potentials, one with vertex at $y = 0$ and equation $4\alpha^4 y^2 + 2\alpha^2$ and two with vertices at $y = \pm \frac{\alpha\sqrt{2}}{\varepsilon}$ and equations $16\alpha^4 (y \mp \frac{\alpha\sqrt{2}}{\varepsilon})^2 - 4\alpha^2$.

Let $0 < \bar{n}_1(\varepsilon) < \frac{\alpha\sqrt{6}}{3\varepsilon}$, $0 < \bar{n}_2(\varepsilon) < \frac{1}{\sqrt{2\nu}}$ with

$$\lim_{\varepsilon\to 0} \bar{\eta}_1(\varepsilon) = \lim_{\varepsilon\to 0} \bar{\eta}_2(\varepsilon) = \infty \tag{3.5}$$

$$\lim_{\varepsilon\to 0} \varepsilon\bar{\eta}_1(\varepsilon) = \lim_{\varepsilon\to 0} \varepsilon\bar{\eta}_2(\varepsilon) = 0 \tag{3.6}$$

Given $\bar{\eta}_1(\varepsilon)$ we choose $\bar{\eta}_2(\varepsilon)$ to be the smallest solution of

$$V_{2\varepsilon}(\bar{\eta}_1(\varepsilon)) = V_{2\varepsilon}\left(\frac{\alpha\sqrt{2}}{\varepsilon} - \bar{\eta}_2(\varepsilon) \right) \tag{3.7}$$

A straightforward computation shows that (3.7) can be solved and that $\bar{\eta}_1(\varepsilon)$ should be of the same order of $\bar{\eta}_2(\varepsilon)$ for $\varepsilon \to 0$.

Let

$$I_1^{(\varepsilon)} = \{y \in \mathbb{R} | \bar{\eta}_1(\varepsilon) < y < \frac{\alpha\sqrt{2}}{\varepsilon} - \bar{\eta}_2(\varepsilon)\} \tag{3.8}$$

$$I_2^{(\varepsilon)} = \{y \in \mathbb{R} | -\frac{\alpha\sqrt{2}}{\varepsilon} + \bar{\eta}_2(\varepsilon) < y < -\bar{\eta}_1(\varepsilon)\} \tag{3.9}$$

we define

$$V_{1\varepsilon}(y) = \begin{cases} V_{2\varepsilon}(y) & \text{when } y \notin I_1^{(\varepsilon)} \cup I_2^{(\varepsilon)} \\ V_{2\varepsilon}(\bar{\eta}_1(\varepsilon)) & \text{when } y \in I_1^{(\varepsilon)} \cup I_2^{(\varepsilon)} \end{cases} \tag{3.10}$$

(see Fig. 6)

Note that $V_{1\varepsilon}$ is a continuous function because of equation (3.7), and as $\varepsilon \to 0$ $V_{1\varepsilon}$ is an approximation to V_ε in the same sense as $V_{2\varepsilon}$.

Let us now consider the operators

$$M_\varepsilon^{(2)} = -\frac{d^2}{dy^2} + V_{2\varepsilon} \qquad y \in \mathbb{R} \tag{3.11}$$

$$M_\varepsilon^{(1)} = -\frac{d^2}{dy^2} + V_{1\varepsilon} \qquad y \in \mathbb{R} \tag{3.12}$$

we will use them to approximate M_ε.

The eigenvalue problem for $M_\varepsilon^{(2)}$

$$M_\varepsilon^{(2)} v = \lambda v \qquad y \in \mathbb{R}, \quad v \in L^2(\mathbb{R}) \tag{3.13}$$

can be reduced to the following eigenvalues problems:

$$M_\varepsilon^{(2)} v = \lambda v \qquad y \in A_+, \quad v \in L^2(A_+) \tag{3.14}$$

$$M_\varepsilon^{(2)} v = \lambda v \qquad y \in A_0, \quad v \in L^2(A_0) \tag{3.15}$$

$$M_\varepsilon^{(2)} v = \lambda v \qquad y \in A_-, \quad v \in L^2(A_-) \tag{3.16}$$

The eigenvalue problems (3.14), (3.15), (3.16) can be solved explicitly. In fact the eigenvalues and eigenfunctions of (3.14) and (3.16) are given by [13], [14].

$$\lambda_{n\varepsilon}^{\pm} = 4\alpha^2\left[2n+\gamma - \frac{\alpha^2}{\nu}\right] \qquad n = 0,1,2, \ldots \tag{3.17}$$

$$\phi_{n\varepsilon}^{\pm} = N_{n\varepsilon}\left[2\alpha^2\left(y \mp \frac{\alpha\sqrt{6}}{3\varepsilon}\right)^2\right]^{\frac{2\gamma+1}{4}} \exp\left(-\alpha^2\left(y \mp \frac{\alpha\sqrt{6}}{3\varepsilon}\right)^2\right) L_n^{(\gamma)}\left[2\alpha^2\left(y \mp \frac{\alpha\sqrt{6}}{3\varepsilon}\right)^2\right] \tag{3.18}$$

where $N_{n\varepsilon}$ is a normalization constant and L_n^γ are the generalized Laguerre polynomials and $\phi_{n\varepsilon}^+$ is defined for $y > \frac{\alpha\sqrt{6}}{3\varepsilon}$, $\phi_{n\varepsilon}^-$ is defined $y < -\frac{\alpha\sqrt{6}}{3\varepsilon}$ and

$$\gamma = \frac{1}{2\nu}\sqrt{4\alpha^4 + \nu^2} \tag{3.19}$$

The eigenvalues and eigenfunctions of (3.15) are given by [15]:

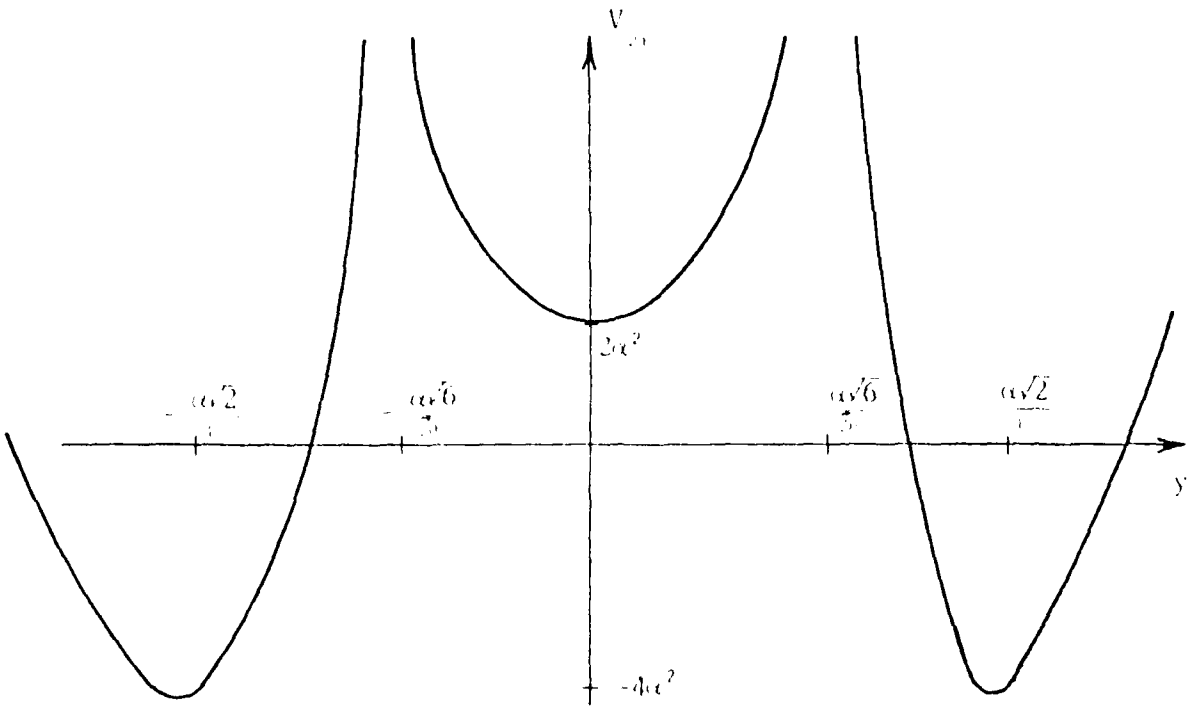

V_{eff}
$2\alpha^2$
$-\frac{\alpha\sqrt{2}}{\gamma}$
$-\frac{\alpha\sqrt{6}}{3\gamma}$
$\frac{\alpha\sqrt{6}}{3\gamma}$
$\frac{\alpha\sqrt{2}}{\gamma}$
y
$-4\alpha^2$

Fig. 5

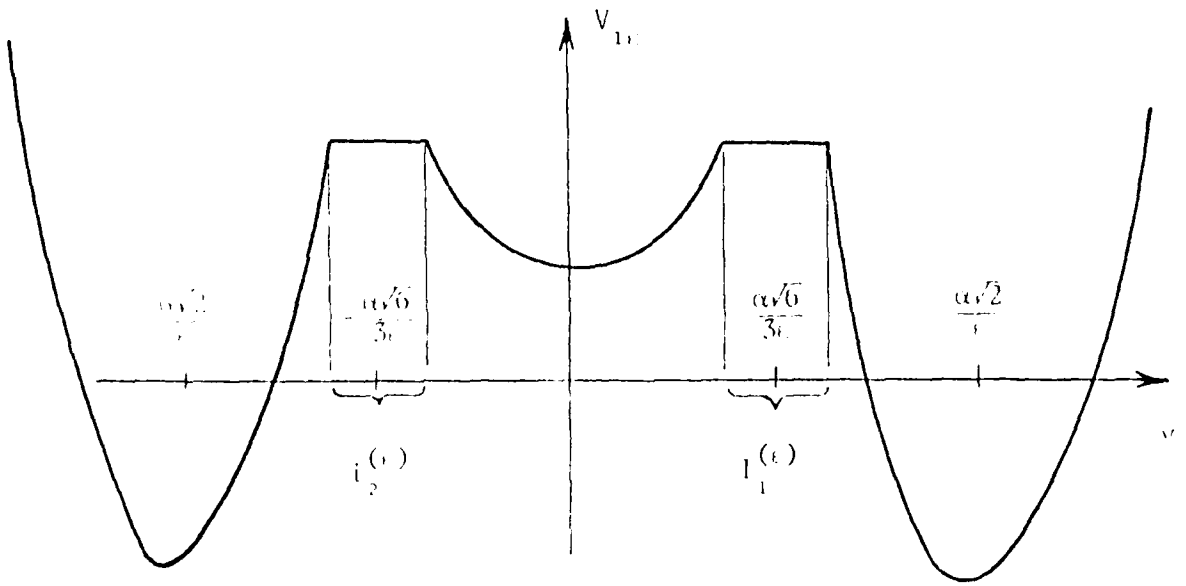

V_{eff}
$-\frac{\alpha\sqrt{6}}{3\gamma}$
$\frac{\alpha\sqrt{6}}{3\gamma}$
$\frac{\alpha\sqrt{2}}{\gamma}$
$I_2^{(\epsilon)}$
$I_1^{(\epsilon)}$
y

Fig. 6

(3.20) $$\lambda^0_{n\varepsilon} = \beta^2[n^2 + \delta(2n+1)] + 2\alpha^2 \qquad n = 0,1, \ldots$$

(3.21) $$\psi^0_{n\varepsilon} = \begin{cases} \cos^{\delta}\beta y\ F(\delta + \frac{n}{2}, \frac{-n}{2}, \frac{1}{2}, \sin^2\beta y) & \text{when } n \text{ is even} \\ \cos^{\delta}\beta y \sin\beta y\ F(\delta + \frac{n+1}{2}, -\frac{n-1}{2}, \frac{3}{2}, \sin^2\beta y) & \text{when } n \text{ is odd} \end{cases}$$

where $F(x_1,x_2,x_3,z)$ is the hypergeometric function and δ is defined by the equation

(3.22) $$V_0 = \beta^2\delta(\delta-1), \qquad \delta > 1.$$

The eigenvalues of (3.13) are given by $\lambda^+_{n\varepsilon}$ and $\lambda^-_{n\varepsilon}$ $n = 0,1,2, \ldots$ the eigenvalues $\lambda^{\pm}_{n\varepsilon}$ have multiplicity two. Moreover as $\varepsilon \to 0$ $\lambda^0_{n\varepsilon}$, $\lambda^{\pm}_{n\varepsilon}$ approaches the eigenvalues (2.24), (2.25), (2.26) of the three harmonic oscillators considered before.

The eigenfunctions of (3.14) satisfy $\psi^+_{n\varepsilon}(\frac{\alpha\sqrt{6}}{3\varepsilon}) = \frac{d\psi^+_{n\varepsilon}}{dy}(\frac{\alpha\sqrt{6}}{3\varepsilon}) = 0$ so that corresponding eigenfunctions of (3.13) can be obtained extending $\psi^+_{n\varepsilon}(y)$ with zero for $y \notin A_+$. Similar statements hold for the eigenfunctions of (3.15), (3.16). Moreover since the eigenfunctions of (3.15) are even or odd and the eigenvalues $\lambda^{\pm}_{n\varepsilon}$ of (3.13) have multiplicity two, the eigenfunction of (3.13) can be chosen to be even or odd.

Let $C^\infty_0(\mathbb{R} - \{\pm\frac{\alpha\sqrt{6}}{3\varepsilon}\}) = \{f : f \text{ is } C^\infty \text{ and of compact support and is zero in a neighborhood of } y = +\frac{\alpha\sqrt{6}}{3\varepsilon} \text{ and } y = -\frac{\alpha\sqrt{6}}{3\varepsilon}\}$. We have:

Theorem 3.3. $M^{(\varepsilon)}_y$ is essentially self adjoint on $C^\infty_0(\mathbb{R} - \{\pm\frac{\alpha\sqrt{6}}{3\varepsilon}\})$.

Proof. It is a straightforward modification of Theorem [2] Appendix 2.

Theorem 3.4. $M_\varepsilon^{(1)}$ is essentially self-adjoint on $C_0^\infty(\mathbb{R})$.

Proof. It follows immediately from Theorem 10.23 page 315 of Weidmann [16].

Let $\bar{A}_+ = \{y | y > \eta_2\}$, $\bar{A}_0 = \{y | 2y_1 - \eta_2 < y < \eta_2\}$, $\bar{A}_- = \{y | y < 2y_1 - \eta_2\}$ and define $U_{2\varepsilon}$ as follows:

$$U_{2\varepsilon} = \begin{cases} \frac{c_2^2}{4}\left(y-\eta_2 - \frac{1}{2\nu_2}\frac{1}{y-\eta_2}\right)^2 - c_2 & \text{when } y \in \bar{A}_+ \\ \frac{\bar{V}_0}{\cos^2\bar{\beta}(y-y_1)} - \bar{V}_0 - c_1 & \text{when } y \in \bar{A}_0 \\ \frac{c_1^2}{4}\left[y-(2y_1-\eta_2) - \frac{1}{2\nu_1}\frac{1}{y-(2y_1-\eta_2)}\right]^2 - c & \text{when } y \in \bar{A}_- \end{cases} \tag{3.23}$$

(see Fig. 7)

where

$$\frac{1}{2\nu_2} = (y_2-\eta_2)^2 \tag{3.24}$$

$$\frac{1}{2\nu_1} = (2y_1-\eta_2)^2 \tag{3.25}$$

$$\bar{\beta} = \frac{\pi}{2}\frac{1}{\eta_2-y_1} \tag{3.26}$$

$$\bar{V}_0 = \frac{c_1^2}{\bar{\beta}^2} \tag{3.27}$$

Let us remember that y_1, y_2, η_1, η_2 depend on ε (Proposition 2.2). It

$$(4.57)\qquad |\hat{V}'''_{1\varepsilon}(y)| \leqslant \frac{1}{4\nu^2}\,\frac{96\alpha^4}{|y - \frac{\alpha\sqrt{6}}{3\varepsilon}|^5} \leqslant 96\alpha^4\,\frac{\sqrt{2\nu}}{|1 - \bar{\eta}_2\sqrt{2\nu}|^5}$$

Since equation (3.7) implies that $\lim_{\varepsilon\to 0}\frac{\bar{\eta}_1(\varepsilon)}{\bar{\eta}_2(\varepsilon)} = \text{constant} \neq 0$ from (4.55), (4.56) and (4.39) we have

$$(4.58)\qquad |F_\varepsilon(y)| \leqslant \text{constant}\ \ \varepsilon^{1-3\delta_1}$$

when $|y - \frac{\alpha\sqrt{2}}{\varepsilon}| < \bar{\eta}_2(\varepsilon)$.

Reasoning in the same way it can be shown that:

$$(4.59)\qquad |F_\varepsilon(y)| \leqslant \text{constant}\ \ \varepsilon^{1-3\delta_1}$$

when $|y + \frac{\alpha\sqrt{2}}{\varepsilon}| < \bar{\eta}_2(\varepsilon)$. This establishes estimate (4.43).

Let us prove (4.44). From Proposition 2.1 (i) we know that $\hat{V}_\varepsilon$ given by (4.27) has three minimizers $y = 0$, $y = \pm\frac{1}{\varepsilon}\left(\frac{\alpha^2 + \sqrt{4\alpha^4+9\varepsilon^2}}{3}\right)^{1/2}$ and two maximizers at $y = \pm\frac{1}{\varepsilon}\left(\frac{4\alpha^2 - \sqrt{4\alpha^4+9\varepsilon^2}}{3}\right)^{1/2}$. Moreover

$$(4.60)\qquad \lim_{\varepsilon\to 0}\varepsilon^2\hat{V}_\varepsilon\left(\pm\frac{1}{\varepsilon}\left(\frac{4\alpha^2 - \sqrt{4\alpha^4+9\varepsilon^2}}{3}\right)^{1/2}\right) = \frac{32}{27}\,\alpha^6$$

$$(4.61)\qquad \lim_{\varepsilon\to 0}\hat{V}_\varepsilon\left(\pm\frac{1}{\varepsilon}\left(\frac{4\alpha^2 + \sqrt{4\alpha^4+9\varepsilon^2}}{3}\right)^{1/2}\right) = -4\alpha^2 + \hat{c}$$

and for $0 < \varepsilon < \varepsilon_0$

$$(4.62)\qquad \frac{1}{\varepsilon}\left(\frac{4\alpha^2 - \sqrt{4\alpha^4+9\varepsilon^2}}{3}\right)^{1/2} \in I_1^{(\varepsilon)}\ ,\quad -\frac{1}{\varepsilon}\left(\frac{4\alpha^2 - \sqrt{4\alpha^4+9\varepsilon^2}}{3}\right)^{1/2} \in I_2^{(\varepsilon)}$$

and

When $|y| < \bar{\eta}_1(\varepsilon) = \varepsilon^{-\delta_1}$ we have

$$(4.52) \qquad |\hat{V}_\varepsilon'''(y)| = |120\varepsilon^4 y^3 - 96\alpha^2\varepsilon^2 y| \leq 24\varepsilon^{2-\delta_1}\,(5^{4-4\delta_1} + 4\alpha^2)$$

and

$$(4.53) \qquad |\hat{V}_{1\varepsilon}'''(y)| = 32\alpha^4\beta|\cos^{-5}\beta y||\sin\beta y|(2+\sin^2\beta y)$$

$$\leq \frac{\varepsilon\ 24\sqrt{6}\ \alpha^3\pi}{|\cos^5(\frac{\pi}{2} - \frac{3}{\alpha\sqrt{6}}\varepsilon^{1-\delta_1})|}$$

So that when $|y| < \bar{\eta}_1(\varepsilon) = \varepsilon^{-\delta_1}$ from (4.51), (4.52), (4.53) we have

$$(4.54) \qquad |F_\varepsilon(y)| \leq \text{constant}\ \ \varepsilon^{1-3\delta_1}$$

When $|y - \frac{\alpha\sqrt{2}}{\varepsilon}| < \bar{\eta}_2(\varepsilon)$ we have $\hat{V}_{1\varepsilon}(y) = \hat{V}_{2\varepsilon}(y)$ so using the Taylor formula at $y = \frac{\alpha\sqrt{2}}{\varepsilon}$ we have

$$(4.55) \qquad F_\varepsilon(y) = -6\alpha\sqrt{2}\ \varepsilon\ (y - \tfrac{\alpha\sqrt{2}}{\varepsilon}) - 3\varepsilon^2(y - \tfrac{\alpha\sqrt{2}}{\varepsilon})^2 + \frac{F_\varepsilon'''(\xi)}{3!}(y - \tfrac{\alpha\sqrt{2}}{\varepsilon})^3$$

with ξ an intermediate point in the interval $\left(\frac{\alpha\sqrt{2}}{\varepsilon}, y.\right)$

For $|y - \frac{\alpha\sqrt{2}}{\varepsilon}| < \bar{\eta}_2(\varepsilon)$ we have:

$$(4.56) \qquad |\hat{V}_\varepsilon'''(y)| \leq 144\sqrt{2}\ \alpha^3 + 624\alpha^2\varepsilon^2\bar{\eta}_2 + 360\sqrt{2}\ \alpha\ \varepsilon^3\bar{\eta}_2^2 + 120\ \varepsilon^4\bar{\eta}_2^3$$

and

Proof: The proof of (4.40) follows from the fact that on $\mathbb{R}\setminus(I_1^{(\varepsilon)} \cup I_2^{(\varepsilon)})$ we have $\hat{V}_{1\varepsilon} = \hat{V}_{2\varepsilon}$. The proof of (4.41) follows from the fact that

$$\hat{V}_{2\varepsilon} \geq \hat{V}_{1\varepsilon} \qquad \text{on} \quad I_1^{(\varepsilon)} \cup I_2^{(\varepsilon)} \tag{4.46}$$

and

$$0 \leq \hat{V}_{2\varepsilon}^{-1}(\hat{V}_{2\varepsilon}-\hat{V}_{1\varepsilon}) = 1 - \frac{\hat{V}_{1\varepsilon}}{\hat{V}_{2\varepsilon}} \leq 1 \tag{4.47}$$

The proof of (4.42) follows from the fact that on $I_1^{(\varepsilon)} \cup I_2^{(\varepsilon)}$ we have

$$\hat{V}_{1\varepsilon}(y) = \hat{V}_{2\varepsilon}(\bar{\eta}_1(\varepsilon)) = V_0 \operatorname{tg}^2\beta\bar{\eta}_1(\varepsilon) + 2\alpha^2 + \hat{c} \tag{4.48}$$

so that since β and V_0 are given by (3.3), (3.4) we have

$$\hat{V}_{1\varepsilon}(y) \geq \text{constant} \quad \varepsilon^{-2\delta_1} \tag{4.49}$$

for $0 < \varepsilon < \varepsilon_0$.

Let us prove (4.43). Let us consider the function

$$F_\varepsilon(y) = \hat{V}_\varepsilon(y) - \hat{V}_{1\varepsilon}(y) \tag{4.50}$$

Using the Taylor's formula at $y = 0$ we have

$$F_\varepsilon(y) = -3\varepsilon^2 y^2 + \frac{F_\varepsilon'''(\xi)}{3!}\, y^3 \tag{4.51}$$

with ξ an intermediate point in the interval (o,y).

Definition 4.5. Let P_2 be the projection on the subspace of the functions of $L^2(\mathbb{R})$ that have support on $\mathbb{R} - U^{(\varepsilon)}$ where

$$U^{(\varepsilon)} = \{y \mid |y| < \bar{\eta}_1(\varepsilon)\} \cup \{y \mid |y - \frac{\alpha\sqrt{2}}{\varepsilon}| < \bar{\eta}_2(\varepsilon)\} \cup \{y \mid |y + \frac{\alpha\sqrt{2}}{\varepsilon}| < \bar{\eta}_2(\varepsilon)\}.$$

That is P_2 is the multiplication operator by $\chi_{\mathbb{R}-U^{(\varepsilon)}}$. Let us now choose

$$\bar{\eta}_1(\varepsilon) = \varepsilon^{-\delta_1} \qquad 0 < \delta_1 < 1/3 \tag{4.39}$$

$\bar{\eta}_2(\varepsilon)$ will remain determined by the equation (3.7).

Theorem 4.6. Let $\bar{\eta}_1(\varepsilon)$ be given by (4.39) and $\bar{\eta}_2(\varepsilon)$ be determined by (3.7). Then for $0 < \varepsilon < \varepsilon_0$ we have the following estimates:

$$\| (\hat{V}_{2\varepsilon} - \hat{V}_{1\varepsilon})(I - P_1) \| = 0 \tag{4.40}$$

$$\| \hat{V}_{2\varepsilon}^{-1} (\hat{V}_{2\varepsilon} - \hat{V}_{1\varepsilon}) P_1 \| \leq \text{constant} \tag{4.41}$$

$$\| \hat{V}_{1\varepsilon}^{-1} P_1 \| \leq \text{constant} \quad \varepsilon^{2\delta_1} \tag{4.42}$$

$$\| (\hat{V}_{\varepsilon} - \hat{V}_{1\varepsilon})(I - P_2) \| \leq \text{constant} \quad \varepsilon^{1-3\delta_1} \tag{4.43}$$

$$\| \hat{V}_{\varepsilon}^{-1} (\hat{V}_{\varepsilon} - \hat{V}_{1\varepsilon}) P_2 \| \leq \text{constant} \tag{4.44}$$

$$\| \hat{V}_{1\varepsilon}^{-1} P_2 \| \leq \text{constant} \quad \varepsilon^{2\delta_1} \tag{4.45}$$

where I is the identity on $L^2(\mathbb{R})$ and $\| \cdot \|$ is the operator norm induced by the L^2 norm.

(4.33) $\qquad (\hat{M}_\varepsilon + z)^2 \geq \hat{V}_\varepsilon^2 \qquad$ on $\quad C_0^\infty(\mathbb{R}) \times C_0^\infty(\mathbb{R})$

(4.34) $\qquad (\hat{M}_\varepsilon^{(1)} + z)^2 \geq \tilde{\beta}\hat{V}_{1\varepsilon}^2 \quad$ on $\quad C_0^\infty(\mathbb{R}) \times C_0^\infty(\mathbb{R})$

(4.35) $\qquad (M_\varepsilon^{(2)} + z)^2 \geq \tilde{\beta}\hat{V}_{2\varepsilon}^2 \quad$ on $\quad C_0^\infty(\mathbb{R} - \{\pm\frac{\alpha\sqrt{6}}{3\varepsilon}\}) \times C_0^\infty(\mathbb{R} - \{\pm\frac{\alpha\sqrt{6}}{3\varepsilon}\})$.

where $\quad 0 < \tilde{\beta} < 1$

Proof: It follows from Theorem 4.1 since $\hat{V}_\varepsilon' = V_\varepsilon'$, $\hat{V}_\varepsilon^2 \geq V_\varepsilon^2$ and $\hat{c} > 0$ and the similar statements for $\hat{V}_{1\varepsilon}$, $V_{1\varepsilon}$, $\hat{V}_{2\varepsilon}$, $V_{2\varepsilon}$.

Theorem 4.3. There exist $z_0 > 0$ and $\varepsilon_0 > 0$ such that for $z \geq z_0$ and $0 < \varepsilon < \varepsilon_0$ we have:

(4.36) $\qquad \| (\hat{M}_\varepsilon + z)^{-1}\psi \| \leq \| \hat{V}_\varepsilon^{-1}\psi \| \qquad \forall \psi \in L^2(\mathbb{R})$

(4.37) $\qquad \| (\hat{M}_\varepsilon^{(1)} + z)^{-1}\psi \| \leq \frac{1}{\tilde{\beta}^{1/2}} \| \hat{V}_{1\varepsilon}^{-1}\psi \| \qquad \forall \psi \in L^2(\mathbb{R})$

(4.38) $\qquad \| (M_\varepsilon^{(2)} + z)^{-1}\psi \| \leq \frac{1}{\tilde{\beta}^{1/2}} \| \hat{V}_{2\varepsilon}^{-1}\psi \| \qquad \forall \psi \in L^2(\mathbb{R})$

Proof: Note that (4.27), (4.28), (4.29) imply $\hat{V}_\varepsilon$, $\hat{V}_{1\varepsilon}$, $\hat{V}_{2\varepsilon} \geq$ constant > 0 so that $\hat{V}_\varepsilon^{-1}$, $\hat{V}_{1\varepsilon}^{-1}$, $\hat{V}_{2\varepsilon}^{-1}$ are bounded operators. The proof of Theorem 4.3 follows immediately from Theorem 2.21, page 330 of Kato [17].

Definition 4.4. Let P_1 be the projection on the subspace of the functions of $L^2(\mathbb{R})$ that have support on $I_1^{(\varepsilon)} \cup I_2^{(\varepsilon)}$. That is P_1 is the multiplication operator given by $\chi_{I_1^{(\varepsilon)} \cup I_2^{(\varepsilon)}}$.

$$(4.26)\quad (1-\chi_{I_1^{(\varepsilon)}\cup I_2^{(\varepsilon)}})\{(1-\bar\beta)V_{2\varepsilon}^2+2zV_{2\varepsilon}+z^2-V_{2\varepsilon}''\}+$$

$$+\chi_{I_1^{(\varepsilon)}\cup I_2^{(\varepsilon)}}\{(1-\bar\beta)V_{2\varepsilon}^2(\bar\eta_1)+2zV_{2\varepsilon}(\bar\eta_1)+z^2\}+$$

$$+\bar c_1\{\delta(y-\bar\eta_1)+\delta(y+\bar\eta_1)\}+\bar c_2\{\delta(y-\frac{\alpha\sqrt{2}}{\varepsilon}+\bar\eta_2)+\delta(y+\frac{\alpha\sqrt{2}}{\varepsilon}-\bar\eta_2)\}\geq 0.$$

In fact for $z > z_0 > 0$, $0 < \varepsilon < \varepsilon_0$ we have $\{(1-\bar\beta)V_{2\varepsilon}^2+2zV_{2\varepsilon}+z^2-V_{2\varepsilon}''\}\geq 0$. Moreover $\{(1-\bar\beta)V_{2\varepsilon}^2(\bar\eta_1)+2zV_{2\varepsilon}(\bar\eta_1)+z^2\}\geq 0$ and $\bar c_1\geq 0$, $\bar c_2\geq 0$.

The estimate (4.2) is established.

Let $\hat c$ be a constant such that

$$(4.27)\qquad \hat V_\varepsilon \equiv V_\varepsilon + \hat c > 0 \quad \text{and} \quad (V_\varepsilon+\hat c)^2 \geq V_\varepsilon^2$$

$$(4.28)\qquad \hat V_{1\varepsilon} \equiv V_{1\varepsilon} + \hat c > 0 \quad \text{and} \quad (V_{1\varepsilon}+\hat c)^2 \geq V_{1\varepsilon}^2$$

$$(4.29)\qquad \hat V_{2\varepsilon} \equiv V_{2\varepsilon} + \hat c > 0 \quad \text{and} \quad (V_{2\varepsilon}+\hat c)^2 \geq V_{2\varepsilon}^2$$

We define

$$(4.30)\qquad \hat M_\varepsilon = -\frac{d^2}{dy^2}+\hat V_\varepsilon = M_\varepsilon + \hat c$$

$$(4.31)\qquad \hat M_\varepsilon^{(1)} = -\frac{d^2}{dy^2}+\hat V_{1\varepsilon} = M_\varepsilon^{(1)} + \hat c$$

$$(4.32)\qquad \hat M_\varepsilon^{(2)} = -\frac{d^2}{dy^2}+\hat V_{2\varepsilon} = M_\varepsilon^{(2)} + \hat c$$

Theorem 4.2. There exist $z_0 > 0$ and $\varepsilon_0 > 0$ such that for $z \geq z_0$ and $0 < \varepsilon < \varepsilon_0$ we have:

The estimate (4.3) has been established.

Let us now prove (4.2). Let $0 < \bar{\beta} < 1$ proceeding as we have done proving (4.1), we obtain as a form on $C_0^\infty(\mathbb{R}) \times C_0^\infty(\mathbb{R})$

$$(4.21)\quad (M_\varepsilon^{(1)}+z)^2 - \bar{\beta}V_{1\varepsilon} \geq (1-\tilde{\beta})V_{1\varepsilon}^2 + 2zV_{1\varepsilon} + z^2 - V_{1\varepsilon}''$$

To prove (4.2) it will be enough to show that for $z > z_0$, $0 < \varepsilon < \varepsilon_0$ we have:

$$(4.22)\quad (1-\tilde{\beta})V_{1\varepsilon}^2 + 2zV_{1\varepsilon} + z^2 - V_{1\varepsilon}'' \geq 0 \qquad y \in \mathbb{R}.$$

Let $\chi_{I_1^{(\varepsilon)} \cup I_2^{(\varepsilon)}}$ be the characteristic function of $I_1^{(\varepsilon)} \cup I_2^{(\varepsilon)}$. We have

$$(4.23)\quad V_{1\varepsilon}'' = V_{2\varepsilon}''(1-\chi_{I_1^{(\varepsilon)} \cup I_2^{(\varepsilon)}}) - \bar{c}_1[\delta(y-\bar{\eta}_1)+\delta(y+\bar{\eta}_1)]$$

$$-\bar{c}_2\left[\delta\left(y - \frac{\alpha\sqrt{2}}{\varepsilon} + \bar{\eta}_2\right) + \delta\left(y + \frac{\alpha\sqrt{2}}{\varepsilon} - \bar{\eta}_2\right)\right]$$

where $\delta(\cdot)$ is the Dirac's delta and

$$(4.24)\quad \bar{c}_1 = 2\beta V_0 \left|\cos^{-3}\beta\bar{\eta}_1 \sin\beta\bar{\eta}_1\right| \geq 0$$

$$(4.25)\quad \bar{c}_2 = \frac{8\alpha^4}{\sqrt{2\nu}}\left|1 - \sqrt{2\nu}\,\bar{\eta}_1 - \frac{1}{(1-\sqrt{2\nu}\,\bar{\eta}_1)^3}\right| \geq 0$$

are the absolute values of the jumps at $y = \pm\,\bar{\eta}_1$ and $y = \pm\left(\frac{\alpha\sqrt{2}}{\varepsilon} - \bar{\eta}_2\right)$ of $V_{1\varepsilon}'$.

Since $V_{1\varepsilon} = V_{2\varepsilon}$ when $y \in \mathbb{R}\setminus\{I_1^{(\varepsilon)} \cup I_2^{(\varepsilon)}\}$ we can rewrite equation (4.22) as follows:

since V_0 given (3.4) goes to infinity when $\varepsilon \to 0$. The last inequality in (4.16) holds $\forall z > 0$, $0 < \varepsilon < \frac{2}{3}\frac{\alpha^2}{\pi}(6(1+\tilde{\beta}))^{\frac{1}{4}}$.

For $y \in A_+$ formula (4.13) becomes:

$$(4.17)\qquad z^2+2\left[4\alpha^4\left(y-\frac{\alpha\sqrt{6}}{3\varepsilon}-\frac{1}{2\nu}\frac{1}{y-\frac{\alpha\sqrt{6}}{3\varepsilon}}\right)^2-4\alpha^2\right]z +$$

$$+(1-\bar{\beta})\left[4\alpha^4\left(y-\frac{\alpha\sqrt{6}}{3\varepsilon}-\frac{1}{2\nu}\frac{1}{y-\frac{\alpha\sqrt{6}}{3\varepsilon}}\right)^2-4\alpha^2\right]^2 +$$

$$-8\alpha^4\left(1+\frac{3}{4\nu^2}\frac{1}{\left(y-\frac{\alpha\sqrt{6}}{3\varepsilon}\right)^4}\right)\geqslant 0$$

with the substitution $t = 2\nu(y-\frac{\alpha\sqrt{6}}{3\varepsilon})^2$ the expression (4.17) becomes:

$$(4.18)\qquad t^2(z^2-8\alpha^2z-8\alpha^4)-24\alpha^4+(1-\bar{\beta})\left[\frac{4\alpha^4}{2\nu}(t-1)^2-4\alpha^2t\right]^2\geqslant 0,\ t\geqslant 0$$

When $t \geqslant \frac{1}{2}$ and z such that $(z^2-8\alpha^2z-8\alpha^4)$ is positive, the left hand side of (4.18) is greater than or equal to

$$(4.19)\qquad \tfrac{1}{4}(z^2-8\alpha^2z-8\alpha^4)-24\alpha^4\geqslant 0 \quad \text{for} \quad z > (4+2\sqrt{30})\alpha^2,\ \varepsilon > 0.$$

When $0 < t < \frac{1}{2}$ and z such that $(z^2-8\alpha^2z-8\alpha^4)$ is positive, then the left hand side of (4.18) is greater than or equal to

$$(4.20)\qquad -24\alpha^4+(1-\bar{\beta})\left[\frac{4\alpha^4}{2\nu}(t-1)^2-4\alpha^2t\right]^2.$$

The expression (4.20) is positive for $0 < \varepsilon < \varepsilon_0$ since ν given by (3.2) goes to zero as $\varepsilon \to 0$.

The proof of (4.3) for $y \in A_-$ is analogous to the proof given for $y \in A_+$ and will be omitted.

and this last expression can be made positive for $z > z_0$ and $0 < \varepsilon < \varepsilon_0$ choosing z_0 and ε_1. The estimate (4.1) is established.

Let us now prove (4.3). Let $0 < \bar{\beta} < 1$ proceeding as we have done proving (4.1), we obtain as a form on $C_0^\infty(\mathbb{R} - \{\pm \frac{\alpha\sqrt{6}}{3\varepsilon}\}) \times C_0^\infty(\mathbb{R} - \{\pm \frac{\alpha\sqrt{6}}{3\varepsilon}\})$

$$(4.12) \qquad (M_\varepsilon^{(2)}+z)^2 - \bar{\beta}V_{2\varepsilon}^2 \geq (1-\tilde{\beta})V_{2\varepsilon}^2 + 2zV_{2\varepsilon} + z^2 - V''_{2\varepsilon}$$

To prove (4.3) it will be enough to show that for $z > z_0$, $0 < \varepsilon < \varepsilon_0$ we have

$$(4.13) \qquad (1-\tilde{\beta})V_{2\varepsilon}^2 + 2zV_{2\varepsilon} + z^2 - V''_{2\varepsilon} \geq 0 \qquad y \in \mathbb{R}.$$

For $y \in A_0$ formula (4.13) becomes:

$$(4.14) \qquad z^2 + 2(V_0 tg^2\beta y+2\alpha^2)z + [(1-\bar{\beta})(V_0 tg^2\beta y+2\alpha^2)^2 + -8\alpha^4 \frac{2\sin^2\beta y+1}{\cos^4\beta y}]$$

when $|y| \leq \frac{\pi}{4\beta}$ we have $\cos^2\beta y \geq \frac{1}{2}$ and $\sin^2\beta y \leq \frac{1}{2}$ so that the expression (4.14) is greater or equal than

$$(4.15) \qquad z^2 + 4\alpha^2 z - (60+4\tilde{\beta}) \geq 0$$

when $z \geq (\sqrt{64+4\tilde{\beta}} - 2)\alpha^2$ and $\forall \varepsilon > 0$.

When $\frac{\pi}{4\beta} \leq |y| < \frac{\pi}{2\beta}$ we have $\sin^2\beta y \geq \frac{1}{2}$ and $\cos^2\beta y \leq \frac{1}{2}$ so that the expression (4.14) is greater or equal than

$$(4.16) \qquad z^2+4\alpha^2 z + \frac{1}{\cos^4\beta y}[(1-\bar{\beta})V_0^2 \sin^4\beta y - 8\alpha^4(2\sin^2\beta y+1)] \geq$$

$$\geq z^2+4\alpha^2 z +4[\frac{1-\bar{\beta}}{4}V_0^2 - 24\alpha^4] \geq 0$$

(4.7) $$F_1(y^2) = 2zV_\varepsilon + z^2 - V_\varepsilon'' \geq 0 \qquad y \in \mathbb{R}$$

Let us define

$$t = y^2$$

$$A = 2z\varepsilon^4$$

$$B = 8z\alpha^2\varepsilon^2 + 30\varepsilon^4$$

$$C = 2z(4\alpha^4 - 3\varepsilon^2) + 48\alpha^2\varepsilon^2$$

$$D = 4z\alpha^2 + z^2 - 8\alpha^4 + 6\varepsilon^2$$

A simple computation shows that

(4.8) $$F_1(t) = t(At^2 - Bt + C) + D \qquad t \geq 0$$

Let us first note that when $z > 2(\sqrt{3}-1)\alpha^2$ and $0 < \varepsilon < 2\alpha^2 \frac{\sqrt{3}}{3}$ we have A, B, C, D positive. Consider now the parabola

(4.9) $$At^2 - Bt + C$$

since $A > 0$, the parabola (4.9) will have a minimizer at $t_0 = \frac{B}{2A}$ where

(4.10) $$At_0^2 - Bt_0 + C = \frac{4AC - B^2}{4A} = -\frac{\varepsilon^2}{2z}(225\varepsilon^2 + 24\alpha^2 z + 12z^2) \leq 0$$

Moreover the equation $At^2 - Bt + C = 0$ has two real roots:

$$0 < t_1 = \frac{B - \sqrt{B^2 - 4AC}}{2A} < t_2 = \frac{B + \sqrt{B^2 - 4AC}}{2A} < \frac{B}{A}$$

So that $\forall t \geq 0$

(4.11) $$F_1(t) \geq \frac{B}{A}(At_0^2 - Bt_0 + C) + D =$$

$$= z^2 - 20\alpha^2 z - 56\alpha^4 - 84\varepsilon^2 - 630\frac{\alpha^2\varepsilon^2}{z} - \frac{3375}{2}\frac{\varepsilon^4}{z^2}$$

§4. The basic estimates.

We will prove here some estimates that will be used later:

Theorem 4.1. There exist constants $z_0 > 0$, $\varepsilon_0 > 0$ such that when $z \geqslant z_0$ and $0 < \varepsilon < \varepsilon_0$ we have

$$(4.1) \qquad (M_\varepsilon + z)^2 \geqslant V_\varepsilon^2 \qquad \text{on} \quad C_0^\infty(\mathbb{R}) \times C_0^\infty(\mathbb{R})$$

$$(4.2) \qquad (M_\varepsilon^{(1)} + z)^2 \geqslant \tilde{\beta} V_{1\varepsilon}^2 \qquad \text{on} \quad C_0^\infty(\mathbb{R}) \times C_0^\infty(\mathbb{R})$$

$$(4.3) \qquad (M_\varepsilon^{(2)} + z)^2 \geqslant \tilde{\beta} V_{2\varepsilon}^2 \qquad \text{on} \quad C_0^\infty(\mathbb{R} - \{ \pm \tfrac{\alpha\sqrt{6}}{3\varepsilon} \}) \times C_0^\infty(\mathbb{R} - \{ \pm \tfrac{\alpha\sqrt{6}}{3\varepsilon} \})$$

where $0 < \tilde{\beta} < 1$.

Proof: Let us first prove (4.1) and let $p = i \frac{d}{dy}$. Then as a form on $C_0^\infty(\mathbb{R}) \times C_0^\infty(\mathbb{R})$ we have:

$$(4.4) \qquad (M_\varepsilon + z)^2 = (p^2 + V_\varepsilon + z)^2 =$$

$$= p^4 + V_\varepsilon^2 + 2zV_\varepsilon + z^2 + 2p(V_\varepsilon + z)p - V_\varepsilon''$$

Since $V_\varepsilon \geqslant$ constant independent of ε when $0 < \varepsilon < \varepsilon_0$ so

$$(4.5) \qquad p(V_\varepsilon + z)p \geqslant 0 \qquad \text{on} \quad C_0^\infty(\mathbb{R}) \times C_0^\infty(\mathbb{R})$$

for z large enough. From (4.4) and (4.5) we have

$$(4.6) \qquad (M_\varepsilon + z)^2 - V_\varepsilon^2 \geqslant 2zV_\varepsilon + z^2 - V_\varepsilon'' \qquad \text{on} \quad C_0^\infty(\mathbb{R}) \times C_0^\infty(\mathbb{R})$$

To prove (4.1) it will be enough to show that for $z \geqslant z_0$ and $0 < \varepsilon < \varepsilon_0$ we have:

Proceeding as before let us now consider

$$N_\varepsilon^{(2)} = -\frac{d^2}{dy^2} + U_{2\varepsilon} \qquad y \in \mathbb{R} \tag{3.32}$$

$$N_\varepsilon^{(1)} = -\frac{d^2}{dy^2} + U_{1\varepsilon} \qquad y \in \mathbb{R} \tag{3.33}$$

we will use them to approximate N_ε.

The eigenvalue problem for $N_\varepsilon^{(2)}$ can be solved analogously to the eigenvalue problem for $M_\varepsilon^{(2)}$ in particular as $\varepsilon \to 0$ the eigenvalues of $N_\varepsilon^{(2)}$ approach the eigenvalues (2.30), (2.31), (2.32) of the three harmonic oscillators considered before.

Let $C_0^\infty(\mathbb{R} - \{2y_1 - n_2\} - \{n_2\}) = \{f | f$ is C^∞ and of compact support and is zero in a neighborhood of $y = 2y_1 - n_2$ and $y = n_2\}$. We have:

Theorem 3.5. $N_\varepsilon^{(2)}$ is essentially self-adjoint on $C_0^\infty(\mathbb{R} - \{2y_1 - n_2\} - \{n_2\})$.

Proof: It is a straightforward modification of Isaacson [2] Appendix 2.

Theorem 3.6. $N_\varepsilon^{(1)}$ is essentially self-adjoint on $C_0^\infty(\mathbb{R})$.

Proof: It follows immediately from Theorem 10.23 page 315 of Weidmann [10].

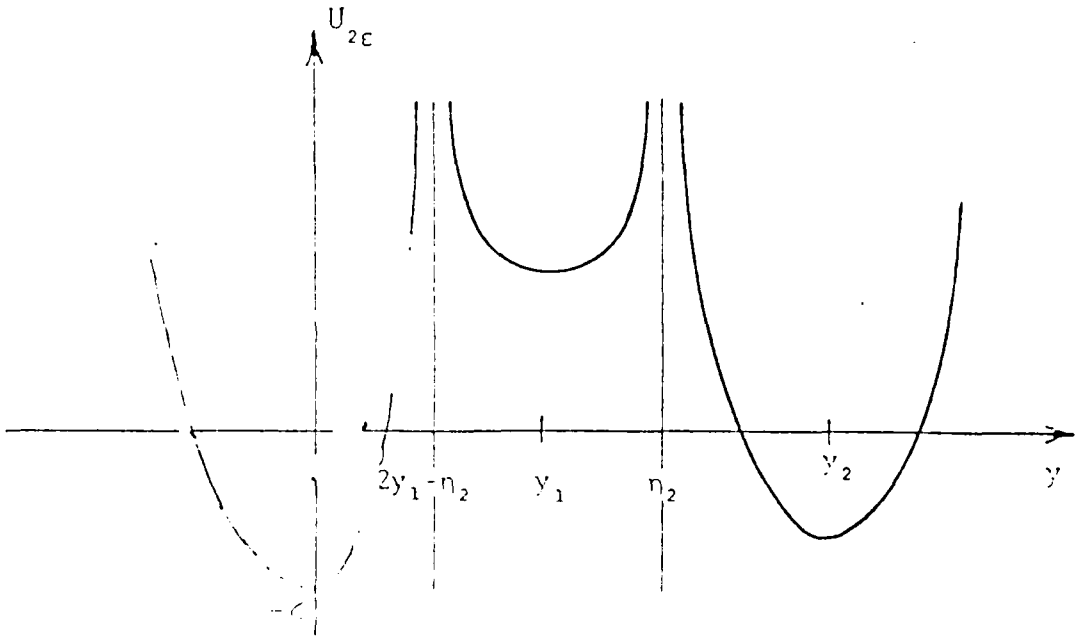

$U_{2\varepsilon}$
$2y_1 - \eta_2$
y_1
η_2
y_2
y

Fig. 7

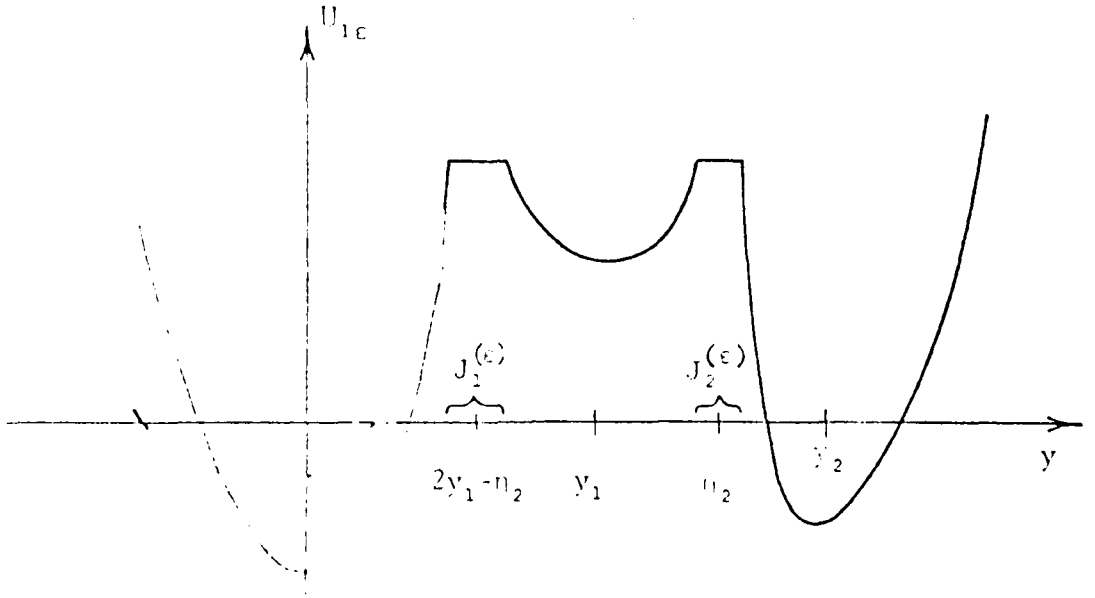

$U_{1\varepsilon}$
$J_1^{(\varepsilon)}$
$J_2^{(\varepsilon)}$
$2y_1 - \eta_2$
y_1
η_2
y_2
y

Fig. 8

is easy to check by explicit computation that $2y_1-\eta_2 > 0$ so that the function $U_{2\varepsilon}$ (Fig. 7) as $\varepsilon \to 0$ is an approximation to U_ε. In particular $U_{2\varepsilon}$ approaches three independent harmonic oscillator potentials one with vertex $y = 0$ and equation c^2y^2-c one with vertex at $y = y_1$ and equation $c_1^2(y-y_1)^2 - c_1$ and one with vertex at $y = y_2$ and equation $c_2^2(y-y_2)^2 - c_2$.

Let $\mu_1(\varepsilon), \mu_2(\varepsilon), \mu_3(\varepsilon) > 0$ and

$J_1^{(\varepsilon)} = \{y\in\mathbb{R} \mid \mu_1(\varepsilon) < y < y_1 - \mu_2(\varepsilon)\}$ and $J_2^{(\varepsilon)} = \{y\in\mathbb{R} \mid y_1 + \mu_2(\varepsilon) < y < y_2-\mu_3(\varepsilon)\}$ two intervals such that

$2y_1-\eta_2 \in J_1^{(\varepsilon)}$ and $\eta_2 \in J_2^{(\varepsilon)}$ such that

$$U_{2\varepsilon}(\mu_1(\varepsilon)) = U_{2\varepsilon}(y_1-\mu_2(\varepsilon)) \tag{3.28}$$

$$U_{2\varepsilon}(y_1+\mu_2(\varepsilon)) = U_{2\varepsilon}(y_2-\mu_3(\varepsilon)) \tag{3.29}$$

and $\overline{J}_1^{(\varepsilon)} \cap \overline{J}_2^{(\varepsilon)} = \{\phi\}$, note that because of symmetry $U_{2\varepsilon}(y_1-\mu_2(\varepsilon)) = U_{2\varepsilon}(y_1+\mu_2(\varepsilon))$

finally later we will need

$$\lim_{\varepsilon\to 0} U_{2\varepsilon}(y_1\mp\mu_2(\varepsilon)) = \infty \ . \tag{3.30}$$

Let us define

$$U_{1\varepsilon}(y) = \begin{cases} U_{2\varepsilon}(y) & y \notin J_1^{(\varepsilon)} \cup J_2^{(\varepsilon)} \\ U_{2\varepsilon}(y_1-\mu_2(\varepsilon)) & y \in J_1^{(\varepsilon)} \\ U_{2\varepsilon}(y_1-\mu_2(\varepsilon)) & y \in J_2^{(\varepsilon)} \end{cases} \tag{3.31}$$

(see Fig. 8)

(4.63) $$\frac{1}{\varepsilon}\left(\frac{4\alpha^2+\sqrt{4\alpha^4+9\varepsilon^2}}{3}\right)^{1/2} \notin I_1^{(\varepsilon)},\ -\frac{1}{\varepsilon}\left(\frac{4\alpha^2+\sqrt{4\alpha^4+9\varepsilon^2}}{3}\right) \notin I_2^{(\varepsilon)}.$$

Let $y \in I_1^{(\varepsilon)} \cup I_2^{(\varepsilon)}$ then $\hat{V}_{1\varepsilon}(y) = V_{2\varepsilon}(\bar{\eta}_1(\varepsilon)) + \hat{c}$ so that

(4.64) $$|\hat{V}_\varepsilon^{-1}(\hat{V}_\varepsilon - \hat{V}_{1\varepsilon})| = |1 - \frac{\hat{V}_{1\varepsilon}}{\hat{V}_\varepsilon}| \leq 1 + \frac{V_{2\varepsilon}(\bar{\eta}_1(\varepsilon)) + \hat{c}}{m(\varepsilon)}$$

where

(4.65) $$m(\varepsilon) = \min_{y \in I_1^{(\varepsilon)} \cup I_2^{(\varepsilon)}} \hat{V}_\varepsilon(y) = \min_{y \in I_1^{(\varepsilon)}} \hat{V}_\varepsilon(y) =$$

$$= \min\{\hat{V}_\varepsilon(\bar{\eta}_1(\varepsilon)),\ \hat{V}_\varepsilon(\frac{\alpha\sqrt{2}}{\varepsilon} - \bar{\eta}_2(\varepsilon))\}$$

and (4.65) follows from the fact that $\hat{V}_\varepsilon$ is even and (4.62), (4.63).

An elementary computation now shows that

(4.66) $$|\hat{V}_\varepsilon^{-1}(\hat{V}_\varepsilon - \hat{V}_{1\varepsilon})| < \text{constant, for } 0 < \varepsilon < \varepsilon_0 \text{ when } y \in I^{(\varepsilon)} \cup I^{(\varepsilon)}.$$

Let $y \geq \frac{\alpha\sqrt{2}}{\varepsilon} + \bar{\eta}_2(\varepsilon)$ we have $V_{1\varepsilon}(y) = V_{2\varepsilon}(y)$. Define $y' = y - \frac{\alpha\sqrt{6}}{3\varepsilon}$ so that we have $y' \geq \bar{\eta}_2(\varepsilon) + \frac{1}{\sqrt{2}y}$ and

(4.67) $$V_{1\varepsilon}(y) = 4\alpha^4(y' - \frac{1}{2\sqrt{2}y'})^2 - 4\alpha^2$$

$$= 4\alpha^4(y' - \frac{1}{\sqrt{2}y})^2(1 + \frac{1}{y'\sqrt{2}y})^2 - 4\alpha^2$$

since when $y' \geq \bar{\eta}_2(\varepsilon) + \frac{1}{\sqrt{2}y}$ we have $(1 + \frac{1}{y'\sqrt{2}y})^2 \leq 4$ it follows

(4.68) $$V_{1\varepsilon}(y) \leq 16\alpha^4(y - \frac{\alpha\sqrt{2}}{\varepsilon})^2 - 4\alpha^2 \text{ when } y \geq \frac{\alpha\sqrt{2}}{\varepsilon} + \bar{\eta}_2(\varepsilon).$$

Moreover

$$(4.69)\qquad V_\varepsilon(y) = -4\alpha^2 - 6\sqrt{2}\,\alpha\varepsilon\left(y - \frac{\alpha\sqrt{2}}{\varepsilon}\right) + (16\alpha^4 - 3\varepsilon^4)\left(y - \frac{\alpha\sqrt{2}}{\varepsilon}\right)^2 +$$

$$+ 24\sqrt{2}\,\alpha^3\varepsilon\left(y - \frac{\alpha\sqrt{2}}{\varepsilon}\right)^3 + 26\,\alpha^2\varepsilon^2\left(y - \frac{\alpha\sqrt{2}}{\varepsilon}\right)^4 +$$

$$+ 6\sqrt{2}\,\alpha\varepsilon^3\left(y - \frac{\alpha\sqrt{2}}{\varepsilon}\right)^5 + \varepsilon^4\left(y - \frac{\alpha\sqrt{2}}{\varepsilon}\right)^6$$

From (4.68), (4.69) when $0 < \varepsilon < \varepsilon_0$ and $y \geq \frac{\alpha\sqrt{2}}{\varepsilon} + \bar{\eta}_2(\varepsilon)$ we have:

$$(4.70)\qquad \left| \frac{V_{1\varepsilon}}{V_\varepsilon} \right| \leq \frac{16\alpha^4 + \frac{4}{\bar{\eta}_2^2(\varepsilon)}}{\left| -\frac{4\alpha^2}{\bar{\eta}_2^2(\varepsilon)} - \frac{6\sqrt{2}\alpha\varepsilon}{\bar{\eta}_2(\varepsilon)} + 16\alpha^2 - 3\varepsilon^2 \right|}$$

so that $\left| \frac{V_{1\varepsilon}}{V_\varepsilon} \right| \leq$ constant when $0 < \varepsilon < \varepsilon_0$. That is

$$(4.71)\qquad |\hat{V}_\varepsilon^{-1}(\hat{V}_\varepsilon - \hat{V}_{1\varepsilon})| \leq \text{constant} \qquad \text{when } 0 < \varepsilon < \varepsilon_0,\ y > \frac{\alpha\sqrt{2}}{\varepsilon} + \eta_2(\varepsilon).$$

Reasoning in the same way it can be shown that

$$(4.72)\qquad |\hat{V}_\varepsilon^{-1}(\hat{V}_\varepsilon - \hat{V}_{1\varepsilon})| < \text{constant} \qquad \text{when } 0 < \varepsilon < \varepsilon_0,\ y < -\frac{\alpha\sqrt{2}}{\varepsilon} - \bar{\eta}_2(\varepsilon).$$

The equations (4.66), (4.71), (4.72) establish (4.44).

Let us prove (4.45). When $y \in I_1^{(\varepsilon)} \cup I_2^{(\varepsilon)}$ we have

$$(4.73)\qquad \hat{V}_{1\varepsilon}(y) = V_{2\varepsilon}(\bar{\eta}_1(\varepsilon)) + c$$

and

$$\lim_{\varepsilon \to 0} \bar{\eta}_1^{-2}(\varepsilon)\, V_{2\varepsilon}(\bar{\eta}_1(\varepsilon)) = \text{constant} \neq 0 \quad . \tag{4.74}$$

From (4.39) it follows that:

$$|\hat{V}_{1\varepsilon}^{-1}| \leqslant \text{constant}\ \ \varepsilon^{2\delta_1} \qquad \text{when} \quad 0 < \varepsilon < \varepsilon_0, \quad y \in I_1^{(\varepsilon)} \cup I_2^{(\varepsilon)}. \tag{4.75}$$

Moreover

$$V_{1\varepsilon}(y) = V_{2\varepsilon}(y) \geqslant V_{2\varepsilon}\left(\frac{\alpha\sqrt{2}}{\varepsilon} + \bar{\eta}_2(\varepsilon)\right) \text{ when } \quad 0 < \varepsilon < \varepsilon_0, \quad y > \frac{\alpha\sqrt{2}}{\varepsilon} + \bar{\eta}_2(\varepsilon) \tag{4.76}$$

and

$$\lim_{\varepsilon \to 0} \bar{\eta}_2^{-2}(\varepsilon)\, V_{2\varepsilon}\left(\frac{\alpha\sqrt{2}}{\varepsilon} + \bar{\eta}_2(\varepsilon)\right) = \text{constant} \neq 0 \tag{4.77}$$

since $\bar{\eta}_1(\varepsilon)$, $\bar{\eta}_2(\varepsilon)$ are of the same order as $\varepsilon \to 0$ from (4.76), (4.77) it follows

$$|\hat{V}_{1\varepsilon}^{-1}| < \text{constant}\ \ \varepsilon^{2\delta_1} \quad \text{when} \quad 0 < \varepsilon < \varepsilon_0, \quad y > \frac{\alpha\sqrt{2}}{\varepsilon} + \bar{\eta}_2(\varepsilon) \tag{4.78}$$

Reasoning in the same way it can be shown that

$$|\hat{V}_{1\varepsilon}^{-1}| < \text{constant}\ \ \varepsilon^{2\delta_1} \quad \text{when} \quad 0 < \varepsilon < \varepsilon_0, \quad y < -\frac{\alpha\sqrt{2}}{\varepsilon} - \bar{\eta}_2(\varepsilon). \tag{4.79}$$

The equations (4.75), (4.78), (4.79) establish (4.45).

This completes the proof of Theorem 4.6.

Theorem 4.7. There exist constants $z_0 > 0$, $\varepsilon_0 > 0$ such that when $z > z_0$ and $0 < \varepsilon < \varepsilon_0$ we have:

$$(4.80) \qquad (N_\varepsilon + z)^2 \geq U^2 \qquad \text{on} \quad C_0^\infty(\mathbb{R}) \times C_0^\infty(\mathbb{R})$$

$$(4.81) \qquad (N_\varepsilon^{(1)} + z)^2 \geq \tilde{\beta}\, U_{1\varepsilon}^2 \qquad \text{on} \quad C_0^\infty(\mathbb{R}) \times C_0^\infty(\mathbb{R})$$

$$(4.82) \qquad (N_\varepsilon^{(2)} + z)^2 \geq \tilde{\beta}\, U_{2\varepsilon}^2 \quad \text{on} \quad C_0^\infty(\mathbb{R} - \{2y_1 - \eta_2\} - \{\eta_2\}) \times C_0^\infty(\mathbb{R} - \{2y_1 - \eta_2\} - \{\eta_2\})$$

where $0 < \tilde{\beta} < 1$.

Proof: Let us first prove (4.80). Proceeding as in the proof of (4.1) we can show that

$$(4.83) \qquad (N_\varepsilon + z)^2 - U_\varepsilon^2 \geq 2zU_\varepsilon + z^2 - U_\varepsilon'$$

So that to prove (4.80) it will be enough to show that for $z > z_0$, $0 < \varepsilon < \varepsilon_0$ we have

$$(4.84) \qquad F_2(y) = 2zU_\varepsilon + z^2 - U_\varepsilon' \geq 0 \qquad y \in \mathbb{R}$$

A simple computation shows that

$$(4.85) \quad F_2(y) = 2a^2\varepsilon^4 zy^6 + \frac{6ab}{\sqrt{2}}\,\varepsilon^3 zy^5 + \{2z(\frac{9}{8}\,b^2 + 2ac)\varepsilon^2 - 30a^2\varepsilon^4\}y^4 +$$

$$+ \{\frac{6}{\sqrt{2}}\,bc\varepsilon z - \frac{60ab}{\sqrt{2}}\,\varepsilon^3\}y^3 + \{2z(c^2 - 3a\varepsilon^2) - 12(\frac{9}{8}\,b^2 + 2ac)\varepsilon^2\}y^2 +$$

$$- \{\frac{6b}{\sqrt{2}}\,\varepsilon z + \frac{18bc}{\sqrt{2}}\,\varepsilon\}y - 2zc - 2c^2 + 6a\varepsilon^2 + z^2$$

Let $t = \varepsilon y$, rearranging the terms in (4.85) we have

$$(4.86)\quad F_2(\tfrac{t}{\varepsilon}) = 2z\{[\tfrac{1}{\varepsilon^2}\tfrac{1}{4}t^2(2at^2+\tfrac{3b}{\sqrt{2}}t+2c)^2 - (3at^2+\tfrac{3b}{\sqrt{2}}t+c)+\tfrac{z}{2}] +$$

$$-\tfrac{1}{2z}[30a^2t^4+\tfrac{60ab}{\sqrt{2}}t^3+12(\tfrac{9}{8}b^2+2ac)t^2+\tfrac{18}{\sqrt{2}}bct+2c^2-6a\varepsilon^2]\}$$

For $z > z_0$ and $0 < \varepsilon < \varepsilon_0$ the expression (4.86) will be positive for any $t \in \mathbb{R}$. This proves (4.80).

The proof of (4.81), (4.82) can be obtained from the proof of (4.2), (4.3) with only minor changes and will be omitted.

Let $\hat{c}_*$ be a constant such that

$$(4.87)\qquad \hat{U}_\varepsilon = U_\varepsilon + \hat{c}_* \quad \text{and} \quad (U_\varepsilon + \hat{c}_*)^2 \geq U_\varepsilon^2$$

$$(4.88)\qquad \hat{U}_{1\varepsilon} = U_{1\varepsilon} + \hat{c}_* \quad \text{and} \quad (U_{1\varepsilon} + \hat{c}_*)^2 \geq U_{1\varepsilon}^2$$

$$(4.89)\qquad \hat{U}_{2\varepsilon} = U_{2\varepsilon} + \hat{c}_* \quad \text{and} \quad (U_{2\varepsilon} + \hat{c}_*)^2 \quad U_{2\varepsilon}^2$$

We define

$$(4.90)\qquad \hat{N}_\varepsilon = -\frac{d^2}{dy^2} + \hat{U}_\varepsilon = N_\varepsilon + \hat{c}_*$$

$$(4.91)\qquad \hat{N}_{1\varepsilon} = -\frac{d^2}{dy^2} + \hat{U}_{1\varepsilon} = N_{1\varepsilon} + \hat{c}_*$$

$$(4.92)\qquad \hat{N}_{2\varepsilon} = -\frac{d^2}{dy^2} + \hat{U}_{2\varepsilon} = N_{2\varepsilon} + \hat{c}_*.$$

Theorem 4.8. There exist $z_0 > 0$ and $\varepsilon_0 > 0$ such that for $z > z_0$ and $0 < \varepsilon < \varepsilon_0$ we have

$$(4.93) \qquad (\hat{N}_\varepsilon + z)^2 \geq \hat{U}_\varepsilon^2 \qquad \text{on} \quad C_0^\infty(\mathbb{R}) \times C_0^\infty(\mathbb{R})$$

$$(4.94) \qquad (\hat{N}_\varepsilon^{(1)} + z)^2 \geq \overline{\beta}\, \hat{U}_{1\varepsilon}^2 \qquad \text{on} \quad C(\mathbb{R}) \times C_0^\infty(\mathbb{R})$$

$$(4.95) \qquad (N_\varepsilon^{(2)} + z)^2 \geq \overline{\beta}\, \hat{U}_{2\varepsilon}^2 \qquad \text{on} \quad C_0^\infty(\mathbb{R} - \{2y_1 - \eta_2\}) \times C_0^\infty(\mathbb{R} - \{2y_1 - \eta_2\} - \{\eta_2\})$$

where $0 < \overline{\beta} < 1$.

Proof: It follows from Theorem 4.7 since $\hat{U}_\varepsilon'' = U_\varepsilon''$, $\hat{U}_\varepsilon^2 \geq U_\varepsilon^2$ and $\hat{c}_* > 0$ and the similar statements for $\hat{U}_{1\varepsilon}$, $U_{1\varepsilon}$, $\hat{U}_{2\varepsilon}$, $U_{2\varepsilon}$.

Theorem 4.9. There exist $z_0 > 0$ and $\varepsilon_0 > 0$ such that for $z \geq z_0$ and $0 < \varepsilon < \varepsilon_0$ we have:

$$(4.96) \qquad \| (\hat{N}_\varepsilon + z)^{-1} \psi \| \leq \| \hat{U}^{-1} \psi \| \qquad \forall \psi \in L^2(\mathbb{R})$$

$$(4.97) \qquad \| (\hat{N}_\varepsilon^{(1)} + z)^{-1} \psi \| \leq \frac{1}{\overline{\beta}^{\frac{1}{2}}} \| \hat{U}_{1\varepsilon}^{-1} \psi \| \qquad \forall \psi \in L^2(\mathbb{R})$$

$$(4.98) \qquad \| (\hat{N}_\varepsilon^{(2)} + z)^{-1} \psi \| \leq \frac{1}{\overline{\beta}^{\frac{1}{2}}} \| \hat{U}_{2\varepsilon}^{-1} \psi \| \qquad \forall \psi \in L^2(\mathbb{R})$$

where $0 < \overline{\beta} < 1$.

Proof: It follows immediately from Theorem 2.21, page 330 of Kato [17].

Definition 4.10. Let P_1^* be the projection on the subspace of the functions of $L^2(\mathbb{R})$ that have support on $J_1^{(\varepsilon)} \cup J_2^{(\varepsilon)}$.

Definition 4.11. Let P_2^* be the projection on the subspace of the functions of $L^2(\mathbb{R})$ that have support on $\mathbb{R}\backslash U_*^{(\varepsilon)}$ where

$$U_*^{(\varepsilon)} = \{y \mid |y| < \mu_1(\varepsilon)\} \cup \{y \mid |y-y_1| < \mu_2(\varepsilon)\} \cup \{y \mid |y-y_2| < \mu_3(\varepsilon)\}$$

Let us now choose

$$\mu_1(\varepsilon) = \varepsilon^{-\delta_1} \qquad 0 < \delta_1 < 1/3 \tag{4.99}$$

$\mu_2(\varepsilon)$ and $\mu_3(\varepsilon)$ will remain determined by the equations (3.28), (3.29).

Theorem 4.12. Let $\mu_1(\varepsilon)$ be given by (4.99) and $\mu_2(\varepsilon)$, $\mu_3(\varepsilon)$ be determined by (3.28), (3.29). Then for $0 < \varepsilon < \varepsilon_0$ we have the following estimates:

$$\|(\hat{U}_{2\varepsilon} - \hat{U}_{1\varepsilon})(I-P_1^*)\| = 0 \tag{4.100}$$

$$\|\hat{U}_{2\varepsilon}^{-1}(\hat{U}_{2\varepsilon}-\hat{U}_{1\varepsilon})P_1^*\| \leq \text{constant} \tag{4.101}$$

$$\|\hat{U}_{1\varepsilon}^{-1}P_1^*\| \leq \text{constant}\ \varepsilon^{2\delta_1} \tag{4.102}$$

$$\|(\hat{U}_{\varepsilon}-\hat{U}_{1\varepsilon})(I-P_2^*)\| \leq \text{constant}\ \varepsilon^{1-3\delta_1} \tag{4.103}$$

$$\|\hat{U}_{\varepsilon}^{-1}(\hat{U}_{\varepsilon}-\hat{U}_{1\varepsilon})P_2^*\| \leq \text{constant} \tag{4.104}$$

$$\|\hat{U}_{1\varepsilon}^{-1}P_2^*\| \leq \text{constant}\ \varepsilon^{2\delta_1} \tag{4.105}$$

Proof: The estimates (4.100), (4.101), ... , (4.105) can be proved has the corresponding estimates (4.40), (4.41), ... , (4.45) of Theorem 4.6.

§5. The behavior as $\varepsilon \to 0$ of eigenvalues and eigenvectors of M_ε, N_ε.

Let us first make precise in which sense $\hat{M}_\varepsilon$ is approximated by $\hat{M}_\varepsilon^{(1)}$, $\hat{M}_\varepsilon^{(2)}$ and $\hat{N}_\varepsilon$ is approximated by $\hat{N}_\varepsilon^{(1)}$, $\hat{N}_\varepsilon^{(2)}$.

Theorem 5.1. There exist constants A, $z_0 > 0$, $\varepsilon_0 > 0$, $\delta_1^* > 0$ such that for $z > z_0$, $0 < \varepsilon < \varepsilon_0$ we have

$$\| (\hat{M}_\varepsilon^{(2)}+z)^{-1} - (\hat{M}_\varepsilon+z)^{-1} \| \leq A\varepsilon^{\delta_1^*} \tag{5.1}$$

$$\| (\hat{M}_\varepsilon^{(2)}+z)^{-1} - (\hat{M}_\varepsilon^{(1)}+z)^{-1} \| \leq A\varepsilon^{\delta_1^*} \tag{5.2}$$

$$\| (\hat{M}_\varepsilon^{(1)}+z)^{-1} - (\hat{M}_\varepsilon+z)^{-1} \| \leq A\varepsilon^{\delta_1^*} \tag{5.3}$$

$$\| (\hat{N}_\varepsilon^{(2)}+z)^{-1} - (\hat{N}_\varepsilon+z)^{-1} \| \leq A\varepsilon^{\delta_1^*} \tag{5.4}$$

$$\| (\hat{N}_\varepsilon^{(2)}+z)^{-1} - (\hat{N}_\varepsilon^{(1)}+z)^{-1} \| \leq A\varepsilon^{\delta_1^*} \tag{5.5}$$

$$\| (\hat{N}_\varepsilon^{(1)}+z)^{-1} - (\hat{N}_\varepsilon+z)^{-1} \| \leq A\varepsilon^{\delta_1^*} \tag{5.6}$$

Proof: The proof of (5.1), (5.2), (5.3) follows from Theorem 4.3 and Theorem 4.6, reasoning as in Isaacson [2], Theorem 3.1. Similarly, the proof of (5.4), (5.5), (5.6) follows from Theorem 4.9 and Theorem 4.12.

Let us remark that (5.1) and (5.4) say that the resolvent of M_ε converges to the resolvent of $M_\varepsilon^{(2)}$ and the resolvent of N_ε converges to the resolvent of $N_\varepsilon^{(2)}$ as $\varepsilon \to 0$. In section 3 we have

studied the eigenvalues and eigenfunctions of $M_\varepsilon^{(2)}$ and $N_\varepsilon^{(2)}$; here we will see the consequences of (5.1), (5.4) on the eigenvalues and of M_ε, N_ε.

Let $P_\varepsilon(S)$ and $P_\varepsilon^{(2)}(S)$ be the spectral projectors of M_ε and $M_\varepsilon^{(2)}$ associated with the Borel set $S \subset \mathbb{C}$.

The eigenvalues of $M_\varepsilon^{(2)}$ (3.17), (3.20) when $\varepsilon \to 0$ are given by

$$\lambda_{n\varepsilon}^{\pm} = 8\alpha^2 n + O(\varepsilon^2) \qquad n = 0,1,2, \ldots \tag{5.7}$$

$$\lambda_{n\varepsilon}^{0} = 4\alpha^2(n+1) + O(\varepsilon^2) \qquad n = 0,1,2, \ldots \tag{5.9}$$

(see Fig. 9).

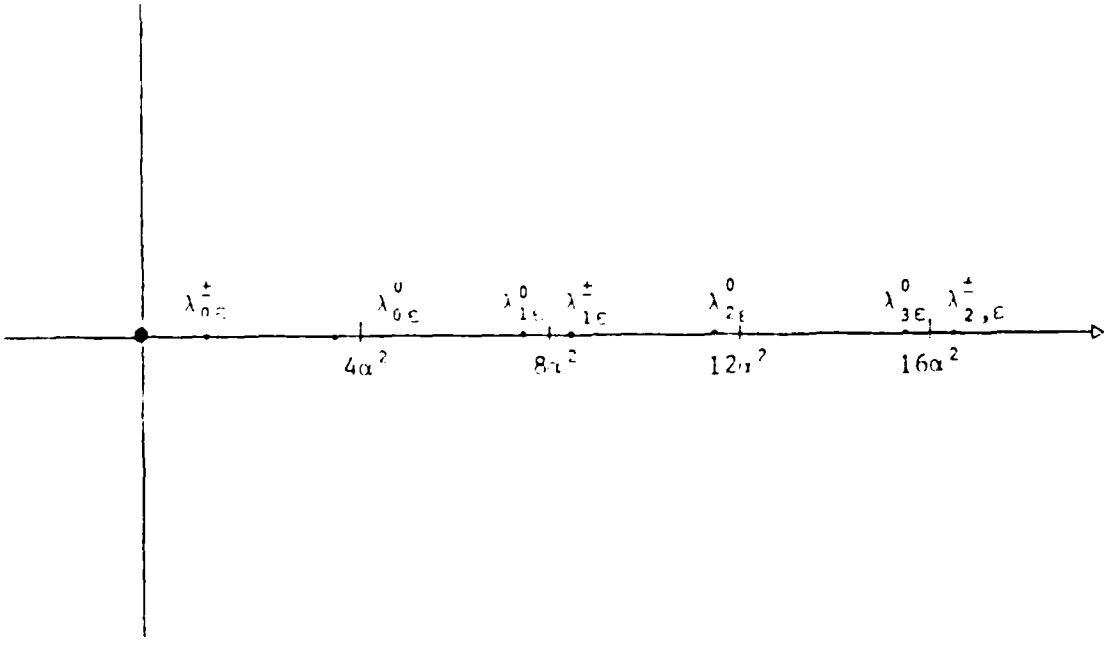


Fig. 9

We remind here that $\lambda_{n\varepsilon}^{\pm}$ has multiplicity 2 and $\lambda_{n\varepsilon}^{0}$ has multiplicity one.

Let

$$C_k(r) = \{z \mid |z-4\alpha^2 k| = r\} \qquad k = 0,1,2, \ldots \tag{5.9}$$

and

$$D_k = \{z \mid |z-4\alpha^2 k| \leq r\} \qquad k = 0,1,2, \ldots \tag{5.10}$$

with $r \leq \alpha^2$, and let:

$$P_\varepsilon^{(2)}(D_k) = \frac{1}{2\pi i} \oint_{C_k(r)} (z-M_\varepsilon^{(2)})^{-1} dz \tag{5.11}$$

then

$$P_\varepsilon^{(2)}(D_0) = P_\varepsilon^{(2)}(\{\lambda_{0\varepsilon}^{\pm}\}) \qquad \text{for } \varepsilon \text{ small enough} \tag{5.12}$$

$$P_\varepsilon^{(2)}(D_k) = \begin{cases} P_\varepsilon^{(2)}(\{\lambda_{k-1,\varepsilon}^{0}\}) & k \text{ odd} \\ P_\varepsilon^{(2)}(\{\lambda_{k-1,\varepsilon}^{0}\} \cup \{\lambda_{\frac{k}{2},\varepsilon}^{\pm}\}) & k \text{ even} \end{cases} \tag{5.13}$$

for $\varepsilon \leq \bar{\varepsilon}_k$ (see Fig. 9). We remark that $\bar{\varepsilon}_k$ cannot be chosen independent of k.

Theorem 5.2. There exists $\bar{\varepsilon}_k > 0$ such that for all $z \in C_k(r)$ and all $0 < \varepsilon < \bar{\varepsilon}_k$.

$$(z - M_\varepsilon)^{-1} \text{ exists} \tag{5.14}$$

$$(5.15)\qquad \sup_{z\in C_k(r)} \| (z-M_\varepsilon)^{-1} - (z-M_\varepsilon^{(2)})^{-1} \| \leq \text{constant}\ \varepsilon^{\frac{\delta_1^*}{2}-}$$

Proof: It follows from Theorem 5.1 and the known properties of the spectrum of $M_\varepsilon^{(2)}$, rearranging the proof of Theorem 4.1 of Isaacson [2].

Theorem 5.3. For $k = 0,1,2, \ldots$ we have

$$\lim_{\varepsilon\to 0} \| P_\varepsilon(D_k) - P_\varepsilon^{(2)}(D_k) \| = 0$$

Moreover for all ε sufficiently small M_ε possesses:

(i) two distinct eigenvalues $\mu_3(\varepsilon) \equiv 0$, $\mu_3'(\varepsilon) > 0$ such that:

$$(5.16)\qquad \lim_{\varepsilon\to 0} \mu_3'(\varepsilon) = \mu_3(\varepsilon) \equiv 0$$

(ii) when k is odd one eigenvalue $\mu_k(\varepsilon)$ such that

$$(5.17)\qquad \lim_{\varepsilon\to 0} \mu_k(\varepsilon) = 4\alpha^2 k \qquad k = 1,3, \ldots$$

(iii) when k is even three distinct eigenvalues $\mu_k(\varepsilon)$, $\mu_k'(\varepsilon)$, $\mu_k''(\varepsilon)$ such that:

$$(5.18)\qquad \lim_{\varepsilon\to 0} \mu_k(\varepsilon) = \lim_{\varepsilon\to 0} \mu_k'(\varepsilon) = \lim_{\varepsilon\to 0} \mu_k''(\varepsilon) = 4\alpha^2 k \qquad k = 2,4, \ldots$$

Proof: From (5.15) of Theorem 5.2 we have:

$$(5.19)\qquad \| P_\varepsilon(D_k) - P_\varepsilon^{(2)}(D_k) \| = \left\| \frac{1}{2\pi i} \oint_{C_k(r)} [(z-M_\varepsilon)^{-1} - (z-M_\varepsilon^{(2)})^{-1}]\,dz \right\|$$

$$\leq \text{constant}\ r^{-\frac{\delta_1^*}{2}}$$

So that for ε sufficiently small

$$\dim P_\varepsilon(D_k) = \dim P_\varepsilon^{(2)}(D_k) \tag{5.20}$$

The remaining part of Theorem 5.3 follows from (5.12), (5.13), (5.7), (5.8).

Let us now establish the results announced in section 2.

Theorem 5.4. Let $0 \equiv -\lambda_0(\varepsilon) < -\lambda_1(\varepsilon) < -\lambda_2(\varepsilon) < \ldots$ be the eigenvalues of M_ε. Then:

$$\lim_{\varepsilon\to 0} -\lambda_1(\varepsilon) = 0 \tag{5.21}$$

$$\lim_{\varepsilon\to 0} -\lambda_{2+4n}(\varepsilon) = 4\alpha^2(2n+1) \qquad n = 0,1,2, \ldots \tag{5.22}$$

$$\lim_{\varepsilon\to 0} -\lambda_{3+4n}(\varepsilon) = \lim_{\varepsilon\to 0} -\lambda_{4+4n}(\varepsilon) = \lim_{\varepsilon\to 0} -\lambda_{5+4n}(\varepsilon) = 8\alpha^2(n+1) \tag{5.23}$$

$$n = 0,1,2, \ldots$$

Proof: Let $S_k = \{z=x+iy \mid -1 \leq x \leq 4\alpha^2 k + 2\alpha^2, -1 \leq y \leq -1\}$ $k = 0,1, \ldots$. By estimates analogous to the ones of Theorem 5.2 it is possible to show that

$$\lim_{\varepsilon\to 0} \| P_\varepsilon(S_k) - P_\varepsilon^{(2)}(S_k) \| = 0$$

That is for ε sufficiently small

$$\dim P_\varepsilon(S_k) = \dim P_\varepsilon^{(2)}(S_k)$$

Theorem 5.4 follows now from Theorem 5.3.

A straightforward computation shows that the eigenvalues of $N_\varepsilon^{(2)}$ when $\varepsilon \to 0$ are given by:

(5.24) $$-\bar{\lambda}_{n\varepsilon}^{(1)} = c(2n+1) - c + O(\varepsilon^2) \qquad n = 0,1,2, \ldots$$

(5.25) $$-\bar{\lambda}_{n\varepsilon}^{(2)} = |c_1|(2n+1) - c_1 + O(\varepsilon^2) \qquad n = 0,1,2, \ldots$$

(5.26) $$-\bar{\lambda}_{n\varepsilon}^{(3)} = c_2(2n+1) - c_2 + O(\varepsilon^2) \qquad n = 0,1,2, \ldots$$

where $c = 2ax_1x_2$, $c_1 = 2ax_1(x_1-x_2) < 0$, $c_2 = 2ax_2(x_2-x_1)$ where x_1, x_2 are given in (i) of Proposition 2.2.

Let $\{-\bar{\lambda}_n\}_{n=0}^{\infty}$ be the set obtained reordering the numbers of $E_1 = \{c(2n+1)-c\}_{n=0}^{\infty}$, $E_2 = \{|c_1|(2n+1)-c_1\}_{n=0}^{\infty}$ and $E_3 = \{c_2(2n+1)-c_2\}_{n=0}^{\infty}$ in such a way that $-\bar{\lambda}_n \leq -\bar{\lambda}_{n+1}$ $n = 0,1, \ldots$. Moreover if a number appears in more than one E_i $i = 1,2,3$ it will appear a corresponding number of times in $\{-\bar{\lambda}_n\}_{n=0}^{\infty}$ in particular since zero appears in E_1 and E_2 we will have $-\bar{\lambda}_0 = -\bar{\lambda}_1 = 0$.

Theorem 5.5. Let $0 \equiv -\bar{\lambda}_0(\varepsilon) \leq -\bar{\lambda}_1(\varepsilon) < \ldots$ be the eigenvalues of N_ε. Then

(5.27) $$\lim_{\varepsilon \to 0} -\bar{\lambda}_n(\varepsilon) = -\bar{\lambda}_n$$

Proof: The proof can be obtained from (5.24), (5.25), (5.26) rearranging the proofs of Theorem 5.2, Theorem 5.3, Theorem 5.4.

We remark that when a certain value appears more than once in $\{-\bar{\lambda}_n\}_{n=0}^{\infty}$ this corresponds to asymptotic eigenvalue degeneracy for N_ε.

Since $-\bar{\lambda}_0 = -\bar{\lambda}_1 = 0$ we have

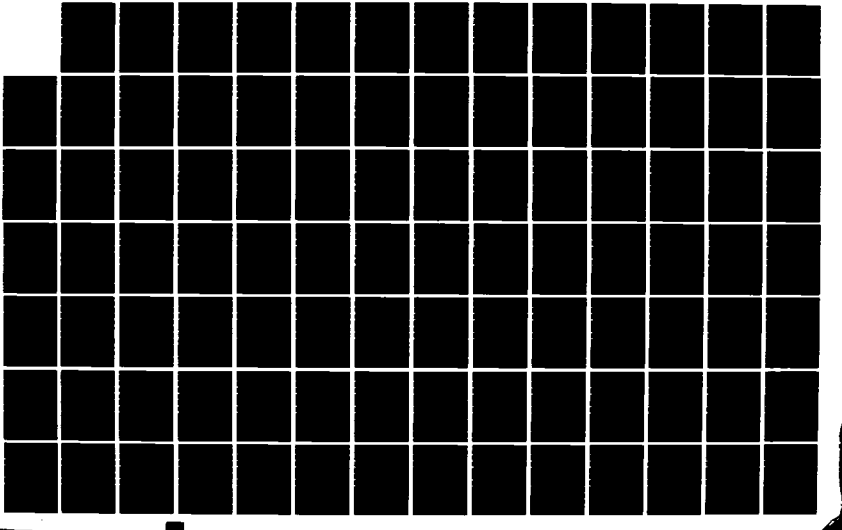
$$\lim_{\varepsilon \to 0} -\bar{\lambda}_1(\varepsilon) = -\bar{\lambda}_0(\varepsilon) \equiv 0$$

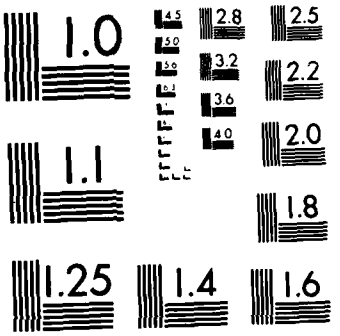
All the remaining $\{-\overline{\lambda}_n\}_{n=2}^{\infty}$ are distinct if $\frac{x_1}{x_2}$ is irrational, if $\frac{x_1}{x_2}$ is rational $\{-\overline{\lambda}_n\}_{n=2}^{\infty}$ contains values that appear only once and values that appear three times.

That is, there are eigenvalues of N_ε that remain isolated when $\varepsilon \to 0$ and eigenvalues that have asymptotic multiplicity three when $\varepsilon \to 0$. We have already observed this phenomenon in the study of M_ε.

AD-A157 546 A NEW METHOD FOR GLOBAL OPTIMIZATION BASED ON STOCHASTIC DIFFERENTIAL EQUATIONS(U) CAMERINO UNIV (ITALY) MATHEMATICS INST F ALUFFI-PENTINI ET AL. 2/3

UNCLASSIFIED DEC 84 DAJA37-81-C-0740 F/G 12/1 NL





MICROCOPY RESOLUTION TEST CHART

§6. The estimate of the first nonzero eigenvalue of M_ε and N_ε.

In section 5 it has been shown that

$$-\lambda_0(\varepsilon) = -\bar{\lambda}_0(\varepsilon) \equiv 0 \qquad \forall \varepsilon \neq 0 \tag{6.1}$$

$$\lim_{\varepsilon \to 0} -\lambda_1(\varepsilon) = \lim_{\varepsilon \to 0} -\bar{\lambda}_1(\varepsilon) = 0 \tag{6.2}$$

where $-\lambda_0(\varepsilon)$, $-\lambda_1(\varepsilon) > 0$ are the first two eigenvalues of M_ε and $-\bar{\lambda}_0(\varepsilon)$, $-\bar{\lambda}_1(\varepsilon) > 0$ are the first two eigenvalues of N_ε.

In section 2 it has been shown that the eigenfunctions corresponding to $-\lambda_0(\varepsilon)$ and $-\bar{\lambda}_0(\varepsilon)$ are respectively:

$$v_0(y) = d_\varepsilon^{\frac{1}{2}} e^{-f_{1\varepsilon}/2} \tag{6.3}$$

and

$$\bar{v}_0(y) = \bar{d}_\varepsilon^{\frac{1}{2}} e^{-f_{2\varepsilon}/2} \tag{6.4}$$

where f_1, f_2 are given by (2.13), (2.14), $f_{1\varepsilon}$, $f_{2\varepsilon}$ by (2.4) and

$$d_\varepsilon = \left(\int_{-\infty}^{+\infty} e^{-f_{1\varepsilon}/2} \, dy \right)^{-1} = \frac{\varepsilon}{\sqrt{2}} c_\varepsilon \tag{6.5}$$

$$\bar{d}_\varepsilon = \left(\int_{-\infty}^{+\infty} e^{-\bar{f}_{2\varepsilon}/2} \, dy \right)^{-1} = \frac{\varepsilon}{\sqrt{2}} \bar{c}_\varepsilon \tag{6.6}$$

are normalization constants such that $\| v_0 \|_{L^2(\mathbb{R})} = \| \bar{v}_0 \|_{L^2(\mathbb{R})} = 1$ and c_ε, $\bar{c}_\varepsilon$ are given by (2.11).

Using the Rayleigh-Ritz principle ([6], page 78, Theorem XIII.2) we want to estimate the quantities $-\lambda_1(\varepsilon) + \lambda_0(\varepsilon)$ and $-\bar{\lambda}_1(\varepsilon) + \bar{\lambda}_0(\varepsilon)$ as $\varepsilon \to 0$ that is the first nonzero eigenvalue of M_ε and N_ε.

The same problem for the Fokker-Planck operators corresponding to M_ε and N_ε and for some more general Fokker-Planck operators has been considered by Matkowsky-Schuss in [10].

Matkowsky-Schuss in [10] used the technique of matching asymptotic expansions. The results obtained here using the Rayleigh-Ritz principle are contained in the ones obtained by Matkowsky-Schuss in [10] but are derived in a more elementary way.

Theorem 6.1. Let $-\lambda_0(\varepsilon)$, $-\lambda_1(\varepsilon)$, M_ε be as above. Then as $\varepsilon \to 0$ we have

$$0 < -\lambda_1(\varepsilon) + \lambda_0(\varepsilon) \equiv -\lambda_1(\varepsilon) \leqslant \text{constant } e^{-\frac{2}{\varepsilon^2}\alpha^4} \tag{6.7}$$

Proof: From the Rayleigh-Ritz principle ([6], page 78, Theorem XIII.2) we have

$$0 < -\lambda_1(\varepsilon) + \lambda_0(\varepsilon) \equiv -\lambda_1(\varepsilon) \leqslant \frac{\langle g, M_\varepsilon g\rangle_{L^2(\mathbb{R})}}{\langle g, g\rangle_{L^2(\mathbb{R})}} \tag{6.8}$$

where $g \in L^2(\mathbb{R})$ is any function orthogonal to v_0 (given by (6.3)) that belongs to the domain of M_ε as a form.

Since v_0 is an even function let us choose

$$g = uv_0 \tag{6.9}$$

where $u(y) = -u(-y)$ is an odd function such that $u \in L^\infty(\mathbb{R})$ and $\frac{du}{dy} \in L^\infty(\mathbb{R})$ where $\frac{du}{dy}$ is the distributional derivative of u.

The function g is orthogonal to v_0 and belongs to the form domain of M_ε.

We have

$$(6.10)\qquad \langle g, M_\varepsilon g\rangle_{L^2(\mathbb{R})} = \int_{-\infty}^{+\infty} uv_0\left(-\frac{d^2}{dy^2} + V_\varepsilon\right)uv_0\,dy =$$

$$= \int_{-\infty}^{+\infty} \left\{ \left[\frac{d}{dy}(uv_0)\right]^2 + V_\varepsilon u^2 v_0^2 \right\} dy =$$

$$= \int_{-\infty}^{+\infty} \left\{ \left(\frac{du}{dy}\right)^2 v_0^2 + u^2\left(\frac{dv_0}{dy}\right)^2 + 2u\,\frac{du}{dy}\,v_0\frac{dv_0}{dy} + V_\varepsilon u^2 v_0^2 \right\} dy$$

$$= \int_{-\infty}^{+\infty} \left(\frac{du}{dy}\right)^2 v_0^2\,dy$$

Since

$$(6.11)\qquad \int_{-\infty}^{+\infty} u^2\left(\frac{dv_0}{dy}\right)^2 dy = -\int_{-\infty}^{+\infty} v_0\frac{d}{dy}\left(u^2\,\frac{dv_0}{dy}\right)dy = -\int_{-\infty}^{+\infty}\left\{2u\,\frac{du}{dy}\,v_0\frac{dv_0}{dy} + u^2 v_0\,\frac{d^2v_0}{dy^2}\right\}dy$$

and $M_\varepsilon v_0 = -\dfrac{d^2v_0}{dy^2} + V_\varepsilon v_0 = 0.$

So that

$$(6.12)\qquad 0 < -\lambda_1(\varepsilon) + \lambda_0(\varepsilon) \equiv -\lambda_1(\varepsilon) \leqslant \frac{\displaystyle\int_{-\infty}^{+\infty}\left(\frac{du}{dy}\right)^2 v_0^2\,dy}{\displaystyle\int_{-\infty}^{+\infty} u^2 v_0^2\,dy}$$

Let us choose

$$u(y) = \begin{cases} 1 & y > 1 \\ y & |y| < 1 \\ -1 & y < -1 \end{cases}$$

equation (6.12) becomes

$$0 < -\lambda_1(\varepsilon) + \lambda_0(\varepsilon) = -\lambda_1(\varepsilon) \leqslant \frac{\int_{-1}^{1} v_0^2 \, dy}{\int_{-\infty}^{+\infty} u^2 v_0^2 \, dy} \tag{6.13}$$

Moreover

$$\int_{-1}^{1} v_0^2 dy = d_\varepsilon \int_{-1}^{+1} e^{-f_{1\varepsilon}(y)} \, dy = \frac{\varepsilon}{\sqrt{2}} c_\varepsilon \int_{-1}^{1} e^{-f_{1\varepsilon}(\varepsilon)} dy \tag{6.14}$$

$$\leqslant \frac{\varepsilon}{\sqrt{2}} c_\varepsilon 2 e^{-f_{1\varepsilon}(1)} = \frac{\varepsilon}{\sqrt{2}} c_\varepsilon 2 \, e^{+2\alpha^2} \, e^{-\frac{\varepsilon^2}{2}} \, e^{-\frac{2}{\varepsilon^2}\alpha^4}$$

It can be easily shown that

$$\lim_{\varepsilon \to 0} \varepsilon c_\varepsilon = \sqrt{\frac{2}{\pi}}\,\alpha \tag{6.15}$$

and that since $x = \frac{\varepsilon}{\sqrt{2}} y$

$$(6.16)\qquad \lim_{\varepsilon\to 0}\int_{-\infty}^{+\infty} v_0^2 u^2 dy = \lim_{\varepsilon\to 0}\int_{-\infty}^{+\infty} c_\varepsilon e^{-\frac{2}{\varepsilon^2} f_1(x)}\, u^2\left(\frac{\sqrt{2}}{\varepsilon}x\right)dx = 1$$

In fact in the sense of distribution

$$(6.17)\qquad \lim_{\varepsilon\to 0} c_\varepsilon e^{-\frac{2}{\varepsilon^2} f_1(x)} = \frac{1}{2}(\delta(x-\alpha) + \delta(x+\alpha))$$

where $\delta(\cdot)$ is the Dirac's delta.

Theorem 6.1 now follows from (6.13), (6.14), (6.15), (6.16).

We remark that since $\alpha^4 = f_1(0) - f_1(\alpha)$ the estimate (6.13) agrees with the one of Matkowsky-Schuss [10].

Theorem 6.2. Let $-\bar{\lambda}_0(\varepsilon)$, $-\bar{\lambda}_1(\varepsilon)$, N_ε be as above. Then as $\varepsilon \to 0$ we have

$$(6.18)\qquad 0 < -\bar{\lambda}_1(\varepsilon) + \bar{\lambda}_0(\varepsilon) \equiv -\bar{\lambda}_1(\varepsilon) \leqslant \text{constant}\ e^{-\frac{2}{\varepsilon^2}(f_2(x_1)-f_2(x_2))}$$

where x_1 and x_2 are given in Proposition 2.2 (i) (see Fig. 2).

Proof: Reasoning as in the proof of Theorem 6.1 we have

$$(6.19)\qquad 0 < -\bar{\lambda}_1(\varepsilon) + \bar{\lambda}_0(\varepsilon) \equiv -\bar{\lambda}_1(\varepsilon) \leqslant \frac{\displaystyle\int_{-\infty}^{+\infty}\left(\frac{dh}{dy}\right)^2 \bar{v}_0^2 dy}{\displaystyle\int_{-\infty}^{+\infty} h^2\bar{v}_0^2 dy}$$

where $g = h\,\bar{v}_0 \in L^2(\mathbb{R})$ is a function orthogonal to $\bar{v}_0$ such that $h \in L^\infty(\mathbb{R})$ and $\frac{dh}{dy} \in L^\infty(\mathbb{R})$.

Let us choose

$$h = \bar{u} - \langle \bar{u}\,\bar{v}_0, \bar{v}_0 \rangle_{L^2(\mathbb{R})} \tag{6.20}$$

where

$$\bar{u}(y) = \begin{cases} 1 & y > y_1 + 1 \\ y - y_1 & |y-y_1| < 1 \\ -1 & y < y_1 - 1 \end{cases} \tag{6.21}$$

where $y_1 = \frac{\sqrt{2}}{\varepsilon} x_1$.

Reasoning as in Theorem 6.1 it can be shown that:

$$\int_{-\infty}^{+\infty} \left(\frac{dh}{dy}\right)^2 \bar{v}_0^2\, dy \leqslant \text{constant}\ e^{-\frac{2}{\varepsilon^2} f_2(x_1)} \tag{6.22}$$

Moreover

$$\int_{-\infty}^{+\infty} h^2 \bar{v}_0^2 dy = \int_{-\infty}^{+\infty} \bar{u}^2 \bar{v}_0^2 dy - \left(\int_{-\infty}^{+\infty} \bar{u}\,\bar{v}_0^2 dy \right)^2 \tag{6.23}$$

and

$$\lim_{\varepsilon \to 0} \int_{-\infty}^{+\infty} \bar{u}^2 \bar{v}_0^2 dy = 1 \tag{6.24}$$

$$(6.25) \qquad \lim_{\varepsilon \to 0} \frac{\int_{-\infty}^{+\infty} \bar{u}\, \bar{v}_0^2 \, dy + 1}{e^{-\frac{2}{\varepsilon^2} f_2(x_2)}} = \sqrt{\frac{f_2''(0)}{f_2''(x_2)}}$$

Theorem 6.2 now follows from (6.19), (6.22), (6.23), (6.24), (6.25).

§7. Conclusions

Let $f(x) \in C^3(\mathbb{R})$ be such that $e^{-\frac{2}{\varepsilon^2} f(x)} \in L^1(\mathbb{R})$ $\forall \varepsilon \neq 0$ and suppose that

$$f'(x) = 0 \tag{7.1}$$

has n roots $\beta_1, \beta_2, \ldots, \beta_n$ such that

$$f''(\beta_i) = \alpha_i \neq 0 \quad i = 1,2, \ldots, n \tag{7.2}$$

that is $\beta_1, \beta_2, \ldots, \beta_n$ are non-degenerate minimizers or maximizers of f.

Let

$$L_\varepsilon(\cdot) = \frac{\varepsilon^2}{2} \frac{\partial^2 \cdot}{\partial x^2} + \frac{\partial}{\partial x} \left\{ \frac{df}{dx} \cdot \right\} \tag{7.3}$$

the Fokker-Planck operator associated to f.

Proceeding as in section 2 the study of the spectrum of (7.3) can be reduced to the study of the spectrum of

$$H_\varepsilon = -\frac{d^2}{dy^2} + W_\varepsilon(y) \tag{7.4}$$

on $L^2(\mathbb{R})$ where $W_\varepsilon(y)$ is given by (2.7).

Let $y_i = \frac{\sqrt{2}}{\varepsilon} \beta_i$ $i = 1,2, \ldots, k$ a straightforward computation shows that as $\varepsilon \to 0$ $W_\varepsilon(y)$ approaches n decoupled harmonic oscillators potentials $\frac{1}{2}\alpha_i^2(y-y_i)^2 - \frac{1}{2}\alpha_i$.

So that we expect the spectrum of H_ε to approximate the spectrum of n decoupled harmonic oscillators $-\lambda_k^i = \frac{1}{2}|\alpha_i|(2k+1) + \frac{1}{2}\alpha_i$, $i = 1,2, \ldots n$ and $k = 0,1,2, \ldots$.

In particular if f has m ($<n$) minimizers, that is $\alpha_{i_j} > 0$ $j = 1,2, \ldots, m$ we expect the eigenvalue zero of H_ε (or L_ε) to have asymptotically multiplicity m when $\varepsilon \to 0$.

References

[1] C. J. Thompson, M. Kac: "Phase transition and eigenvalue degeneracy of a one-dimensional anharmonic oscillator" Studies Applied Math. 48, (1969), 257-264.

[2] D. Isaacson: "Singular perturbations and asymptotic eigenvalues degeneracy" Comm. Pure Applied Math. 29, (1976), 531-551.

[3] D. Isaacson, D. Marchesin: "The eigenvalues and eigenfunctions of a spherically symmetric anharmonic oscillator" Comm. Pure Aplied Math. 31, (1978), 659-670.

[4] E. Harrell: "On the rate of asymptotic eigenvalue degeneracy" Comm. Math. Phys. 60, (1978), 73-95.

[5] E. Harrell: "Double wells" Comm. Math. Phys. 75, (1980), 239-261.

[6] M. Reed, B. Simon: "Methods of modern mathematical physics" vol. IV, Academic Press, New York, 1978.

[7] J. Glimm, A. Jaffe: "Quantum physics: a functional integral point of view" Springer-Verlag, New York, 1981.

[8] Z. Schuss: "Singular perturbation methods for stochastic differential equations of mathematical physics" SIAM Rev. 22, (1980), 119-155.

[9] S. Chandresekhar: "Stochastic problems in physics and astronomy" in Selected Papers on Noise and Stocastic Processes, N. Wax Editor, Dover, New York, 1954.

[10] B. J. Matkowsky, Z. Schuss: "Eigenvalues of the Fokker-Planck operator and the approach to equilibrium for diffusions in potential fields" SIAM J. Appl. Math. 40, (1981), 242-254.

[11] F. Aluffi-Pentini, V. Parisi, F. Zirilli: "Global optimization and stochastic differential equations" submitted to Journal of Optimization Theory and Applications.

[12] B. Simon: "Coupling constant analyticity for the anharmonic oscillator" Annals of Physics 58, (1970), 76-136.

[13] F. Zirilli: "Some observations on the operator $H = -\frac{1}{2}\frac{d^2}{dx^2} + m^2x^2 + \frac{g}{x^2}$" Journal Mathematical Physics 15, (1974), 1202.

[14] E. C. Titchmarsh: "Eigenfunction Expansions" Part I, Oxford University Press, London, 1962.

[15] S. Flügge: "Rechenmethoden der Quantentheorie: 1 Teil Elementare Quantenmechanik" Springer-Verlag, Berlin, 1947.

[16] J. Weidmann: "Linear operators in Hilbert spaces" Springer-Verlag, New York, 1980.

[17] T. Kato: "Perturbation theory for linear operators" Springer-Verlag, New York, 1966.

APPENDIX A3

Test problems for global optimization software

by F. Aluffi-Pentini, V. Parisi, F. Zirilli

(submitted to ACM Transactions on Mathematical Software).

Problem 14

A function with three ill-conditioned minima, a = 10^5

a) function: as in probl. 10

b) parameter values: a = 10^5, b = 1/a
c), d), e) : as in problem 10
f) minima in the region D : two global minima at
(x,y) = ±(0, 14.94511), where f = -24776.51834,
and another local minimum at
(x,y) = (0, 0), where f = 0
g), h), i) : as in problem 10.

Problem 15

A function with three ill-conditioned minima, a = 10^6

a) function: as in probl. 10

b) parameter values: a = 10^6, b = 1/a
c), d), e) : as in problem 10
f) minima in the region D : two global minima at
(x,y) = ±(0, 26.58678), where f = -249293.01826
and another local minimum at
(x,y) = (0, 0), where f = 0
g), h), i) : as in problem 10.

Problem 16
Goldstein-Price function

a) function:

$$f(x,y) = g(x,y)\, h(x,y)$$

with $g(x,y) = 1 + u^2 (36 - 20u + 3u^2)$

$h(x,y) = 30 + v^2 (18 - 16v + 3v^2)$

where $u = x + y + 1$, and $v = 2x - 3y$

Problem 11
A function with three ill-conditioned minima, a = 100

a) function: as in probl. 10
b) parameter values: a = 100, b = 1/a
c), d), e) : as in problem 10
f) minima in the region D : two global minima at
(x,y) = ±(0, 2.60891), where f = -18.05870
and another local minimum at
(x,y) = (0, 0), where f = 0
g), h), i) : as in problem 10.

Problem 12
A function with three ill-conditioned minima, a = 1000

a) function: as in probl. 10
b) parameter values: a = 1000, b = 1/a
c), d), e) : as in problem 10
f) minima in the region D : two global minima at
(x,y) = ±(0, 4.70174), where f = -227.76575
and another local minimum at
(x,y) = (0, 0), where f = 0
g), h), i) : as in problem 10.

Problem 13
A function with three ill-conditioned minima, a = 10000

a) function: as in probl. 10
b) parameter values: a = 10000, b = 1/a
c), d), e) : as in problem 10
f) minima in the region D : two global minima at
(x,y) = ±(0, 8.39401), where f = -2429.41477
and another local minimum at
(x,y) = (0, 0), where f = 0
g), h), i) : as in problem 10.

Problem 8
Two-dimensional penalized Shubert function, $\beta = 0.5$

a) function: as in probl. 7
b) parameter values: as in probl. 7, except: $\beta = 0.5$
c), d), e) : as in probl. 7
f) minima in the region D : general behavior as in probl. 7
but 17 out of the 18 global minima become non-global,
ving a single global minimum at
$(x,y) = (-1.42513, -0.80032)$ with the same value for f.
g), h), i) : as in probl. 7.

Problem 9
Two-dimensional penalized Shubert function, $\beta = 1$

a) function: as in probl. 7
b) parameter values: as in probl. 7, except: $\beta = 1$
c), d), e) : as in probl. 7
f) minima in the region D : same behavior as in problem 8
g), h), i) : as in probl. 7.

Problem 10
A function with three ill-conditioned minima, a = 10

a) function:

$$f(x,y) = ax^2 + y^2 - (x^2 + y^2)^2 + b(x^2 + y^2)^4$$

b) parameter values: a = 10, b = 1/a
c) dimension: N = 2
d) region: D = { $|x| \leq 10$, $|y| \leq 100$ }
e) penalization: none
f) minima in the region D : two global minima at
$(x,y) = \pm(0, 1.38695)$, where f = -0.40746
and another local minimum at
$(x,y) = (0, 0)$, where f = 0
g) initial point: $(x_0, y_0) = (0, 0)$
h) source: suggested by one of the authors (F.Z.)
i) notes: as in problem 2; the problem becomes more ill-conditioned as a becomes larger.

Problem 7
Two-dimensional penalized Shubert function, $\beta = 0$

a) function:

$$f(x,y) = g(x)\, g(y) + \beta\, [(x - a)^2 + (y - b)^2]$$

where g is the function defined as f in probl. 3, a)

b) parameter values: $\beta = 0$
$a = -1.4251284283197609708$, $b = -0.80032110047197312466$

c) dimension: $N = 2$

d) region: $D = \{\, |x| \le 10,\ |y| \le 10 \,\}$

e) penalization: $w(x,y) = u(x,10,100,2) + u(y,10,100,2)$

f) minima in the region D : 18 global minima, at

(x,y) = (-7.08350, -7.70831)
(x,y) = (-0.80032, -7.70831)
(x,y) = (5.48286, -7.70831)
(x,y) = (-7.70831, -7.08350)
(x,y) = (-1.42513, -7.08350)
(x,y) = (4.85805, -7.08350)
(x,y) = (-7.08350, -1.42513)
(x,y) = (-0.80032, -1.42513)
(x,y) = (5.48286, -1.42513)
(x,y) = (-7.70831, -0.80032)
(x,y) = (-1.42513, -0.80032)
(x,y) = (4.85805, -0.80032)
(x,y) = (7.08350, 4.85805)
(x,y) = (-8.00320, 4.85805)
(x,y) = (5.48286, 4.85805)
(x,y) = (-7.70831, 5.48286)
(x,y) = (-1.42513, 5.48286)
(x,y) = (4.85805, 5.48286), where f = -186.73091

and 742 other local minima.

g) initial point: $(x_o, y_o) = (0, 0)$

h) source: ref. [5]

i) notes: outside D the penalized function f + w has a small number of non-global minima (near to D); the point (a,b) is one of the 18 global minimizers of g(x) g(y) in the region D .
Three-dimensional plots of f are given in [5].

Problem 5
A function with a single row of local minima

a) function:

$$f(x,y) = ax^2 + (1/2)[1 - \cos(2x)] + y^2$$

b) parameter values: a = 0.05
c) dimension: N = 2
d) region: D = { $-15 \leq |x| \leq 25$, $-5 \leq |y| \leq 15$ }
e) penalization: none
f) minima in the region D : a global minimum at
(x,y) = (0, 0), where f = 0
and six other local minima at:
(x,y) = ±(2.98978, 0), where f = 0.46981
(x,y) = ±(5.96370, 0), where f = 1.97693
(x,y) = ±(8.87846, 0), where f = 4.21128

g) initial point: $(x_0, y_0) = (-3, 0)$

h) source: suggested by one of the authors (F.Z.)
i) notes: the starting point is very close to a non-global minimizer.

Problem 6
Six-hump camel function

a) function:

$$f(x,y) = (4 - 2.1\,x^2 + x^4/3)\,x^2 + xy + (-4 + 4y^2)\,y^2$$

b) parameter values: none
c) dimension: N = 2
d) region: D = { $|x| \leq 3$, $|y| \leq 2$ }
e) penalization: none
f) minima in the region D : two global minima, at
(x,y) = ±(-0.089842, 0.71266), where f = -1.03163
and four other local minima, at
(x,y) = ±(-1.70361, 0.79608), where f = -0.21546, and
(x,y) = ±(1.60710, 0.56865), where f = 2.10425

g) initial point: $(x_0, y_0) = (0, 0)$

h) source: ref. [1], quoted by [5]
i) notes: in D the function f has 2 maxima and 7 saddle-points. Three-dimensional plots of f are given in [5].

Problem 3
One-dimensional penalized Shubert function

a) function:

$$f(x) = \sum_{k=1}^{5} k \cos [(k+1)x + k]$$

b) parameter values: none
c) dimension: N = 1
d) region: D = (|x| ≤ 10)
e) penalization: w(x) = u(x,10,100,2)
f) minima in the region D : three global minima at
x = -7.70831, -1.42513, 4.85806, where f = -12.87088
and 16 other local minima

g) initial point: $x_0 = 0$

h) source: ref. 15 of [5]
i) notes: the function f is periodic (period 2π).

Problem 4
A fourth-order polynomial in two variables

a) function:

$$f(x) = x^4/4 - x^2/2 + ax + y^2/2$$

b) parameter values: a = 0.1
c) dimension: N = 2
d) region: D = (|x| ≤ 10, |y| ≤ 10)
e) penalization: none
f) minima in the region D :
two minima, both for y = 0, as in problem 1

g) initial point: $(x_0, y_0) = (1, 0)$

h), i) : as in probl. 1.

APPENDIX 1. The test-problem list.

Problem 1
A fourth-order polynomial

a) function:

$$f(x) = x^4/4 - x^2/2 + ax$$

b) parameter values: a = 0.1
c) dimension: N = 1
d) region: D = { $|x| \leq 10$ }
e) penalization: none
f) minima in the region D : a global minimum at
x = -1.04668, where f = -0.35239,
and another local minimum at
x = 0.94565, where f = -0.15264

g) initial point: $x_0 = 1$

h) source: suggested by one of the authors (F.Z.)
i) notes: the initial point is very close to the non-global minimizer.

Problem 2
Goldstein sixth-order polynomial

a) function:

$$f(x) = x^6 - 15x^4 + 27x^2 + 250$$

b) parameter values: none
c) dimension: N = 1
d) region: D = { $|x| \leq 4$ }
e) penalization: none
f) minima in the region D : two global minima at
x = ±3, where f = 7,
and another local minimum at
x = 0, where f = 250

g) initial point: $x_0 = 0$

h) source: ref. 6 of [5]
i) notes: the starting point is exactly at the non-global minimizer, midway between the global ones.

References

[1] L. C. W. Dixon, G. P. Szego, (eds.), TOWARDS GLOBAL OPTIMISATION, North-Holland Publ. Co., 1975.

[2] L. C. W. Dixon, G. P. Szego, (eds.), TOWARDS GLOBAL OPTIMISATION 2, North-Holland Publ. Co., 1978.

[3] L. C. W. Dixon, G. P. Szego, "The Global Optimisation Problem", in ref. [2], p. 1-15.

[4] J. Moré, B. Garbow, K. Hillstrom, "Testing unconstrained optimization software", A. C. M. Transactions on Mathematical Software, 7 (1981), 17-41.

[5] A. V. Levy, A. Montalvo, "Algoritmo de tunelización para la optimización global de funciones", Comunicaciones técnicas, Serie Naranja, n. 204, IIMAS-UNAM, México D. F., 1979.

[6] A. V. Levy, A. Montalvo, S. Gómez, A. Calderón, "Topics in global optimization", in: J. P. Hennart (ed.), NUMERICAL ANALYSIS, Lecture Notes in Mathematics n. 909, Springer, 1982, p. 18-33.

[7] F. Aluffi-Pentini, V. Parisi, F. Zirilli, "Global optimization and stochastic differential equations", submitted to Mathematics of Computation.

[8] B. G. Ryder, A. D. Hall, "The PFORT Verifier", Computing Science Technical Report n. 12, Bell Laboratories, Murray Hill, N. J., Jan. 1981.

The subroutine GLOMTF contains a total of about 430 statements (including some 160 comment lines). This amounts on the ASCII FORTRAN compiler (without optimization option, version 10R1A) of the UNIVAC EXEC 8 operating system (level 37R2C) to a storage requirement of about 1040 (36-bit) words for the instructions and about 560 words for the data.

The corresponding approximate data for the subroutine GLOMIP are 330 statements (including 140 comment lines), and 350 and 35 words for instructions and data.

4. Conclusions

We have provided an extensive set of test problems (including the FORTRAN coding) to be used for testing global optzation software.

The prospective user may find it useful to exploit a ready-made selection of fully coded test problems of known properties, which may save him time, effort, and possible coding errors, while enabling a more uniform comparison with the results of other users.

3. The FORTRAN subroutines

The test problems described in sect. 2 have been coded in the form of two FORTRAN subroutines, GLOMIP and GLOMTF .

For a given test problem the subroutine GLOMIP returns the number N of variables, the initial point x_0 , and the observation region D , and the subroutine GLOMTF returns the basic test function f , possibly penalized (outside D) by the penalization function w .

All the coding is written in FORTRAN IV , and meets the specifications of PFORT , a portable subset of A.N.S. FORTRAN (ref. [8]). The FORTRAN implicit type definition for integers is used throughout; all non-integer variables are double-precision.

The call statement of GLOMIP is:

```
CALL GLOMIP (NPROB, N, XO, XMIN, XMAX)
```

where

NPROB is the (input) number which identifies the test problem according to the sequence in Appendix 1

N is the (output) dimension of the problem (number of independent variables)

XO is the (output) N-vector containing the suggested initial point x_0

XMIN and XMAX are (output) N-vectors containing the boundaries of the observation region D , defined by

$$XMIN(I) \leq X(I) \leq XMAX(I) , \quad I = 1,\ldots,N$$

The call statement of GLOMTF is:

```
CALL GLOMTF (NPROB, N, X, FUNZ)
```

where

NPROB is the (input) problem number (see above)

N is the (input) problem dimension (must be equal to the value provided by GLOMIP)

X is the (input) N-vector containing the point x at which the test function is to be computed

FUNZ is the (output) value of the (possibly penalized) test function at x .

Therefore a test problem, such as we provide it here, is defined by:

a basic function f
an observation region D
a penalization function w, if needed ($w = 0$ in D)
an initial point x_0.

We think that such an arrangement covers the needs of a wide spectrum of possible global minimization methods: a truly unconstrained method will try to minimize $f + w$ in R^N, without exploiting any information about D, while a constrained method will try to minimize f in D, obviously ignoring w.

We provide a set of 37 test problems complying with the above format, with varying source, nature, and difficulty. A complete definition of the test problems, together with some relevant information, is reported in Appendix 1, where for each problem we give:

a) basic unrestricted function f
b) numerical values of any parameter in f
c) problem dimension N (number of independent variables)
d) observation region D, which is always in the form of an N-dimensional interval

$$D = \{ Xmin_i \leq x_i \leq Xmax_i , \ i = 1,\ldots,N \}$$

e) penalization function w (if any), which is always defined by means of a standard penalization function u of a single real variable x (with $a > 0$, $b > 0$)

$$u(x,a,b,m) = b (|x| - a)^m \qquad (|x| > a)$$
$$= 0 \qquad (|x| \leq a)$$

f) information about the minima of f in D (location of global minimizers and corresponding function value, given to at least 5 decimal places and at least 5 significant figures, and - whenever possible - analogous information about the local minimizers).
g) initial point x_0
h) source
i) notes, if any.

2. The test problem set

Let us consider the problem of finding a global minimizer of a real-valued function f of N real variables, i.e. a point x^* in R^N such that $f(x) \geq f(x^*)$ for all x in R^N.

In the context of this (unconstrained) global minimization problem to give a test problem simply amounts to give a test function.

It is however a fact that many of the global minimization methods reported in the literature (see for example [1] and [2]) only attempt, by their very nature, to find a global minimum of the function f restricted to a compact region D.

While this may be strictly considered a constrained global minimization problem, the only practical consequence for the test-problem builder is that in order to give a test problem for one of the above methods one must give a test function f together with a compact region D.

As far as the above methods are concerned the behavior of the test function f outside the region D is clearly irrelevant, and may be arbitrary.

Since however our aim is to provide a single set of test problems it is clear that, in order to meaningfully use the above problems also to test and compare the methods attempting to perform a strictly global minimization (see for example [7]), it becomes necessary that the minimization-relevant behavior of f be sufficiently "concentrated" around the region D, i.e. the unrestricted f has all its global minima inside D, at most a small number of local minima outside D and a sufficiently rapid growth away from D.

Since these conditions are not fulfilled by some of the test functions actually proposed by some authors (as - for instance in the interesting "oscillating" problems in [5]), we have adopted the simple solution of "penalizing", whenever needed, the original function f outside D, by simply adding to f a penalization function w which is identically zero in D and of sufficiently rapid growth away from D.

Finally - since some methods need a starting point - we complete our definition of a test problem by providing an initial point x_0.

1. Introduction

The problem of finding a global minimizer of a real-valued function of several real variables is of considerable practical and theoretical interest, and many algorithms for its numerical solution have been developed; see for example the two volumes of collected papers [1] and [2], and the survey paper [3].

The situation appears to be still in a rapidly evolving state, and far from the more mature state reached by the simpler problem of finding a local minimizer.

While this makes it difficult, and perhaps untimely, to attempt a systematic classification of the algorithms, it does not relieve us from the need of testing the current algorithms, both for validation and for comparison.

The experimental testing of the algorithms is usually performed by running their software implementation on a number of test problems; a standard set of test problems is clearly useful, being of verifiable quality, and allowing a fair comparison of the algorithms.

The importance of an extensive testing on a sufficient number of carefully selected test problems has been stressed by Moré & al. [4], in the different context of local minimization.

In the field of global minimization a common set of test problems was agreed upon by many of the authors contributing to [2], and is reported by many of them and in Appendix 1 of [3]. A number of more difficult test problems were used by Levy and Montalvo and are described in [5] and [6].

The present authors have been involved in a global minimization project [7], and in order to test their own algorithms they have made use of a large set of test problems, including those in [3] and [5].

The purpose of this paper is to make generally available the above set of test problems, including their software implementation in the FORTRAN IV programming language.

In section 2 we describe the general pattern of the test problem set, in section 3 we describe the usage of the FORTRAN subroutines implementing the problem set.

A detailed list of the test functions is reported in Appendix 1, while the complete FORTRAN list is in Appendix 2.

TEST PROBLEMS
FOR
GLOBAL OPTIMIZATION SOFTWARE

Filippo Aluffi-Pentini
Dipartimento di Matematica, Università di Bari, 80125 Bari (Italy)

Valerio Parisi
Dipartimento di Fisica, II Università di Roma (Tor Vergata), 00173 Roma (Italy)

Francesco Zirilli
Istituto Matematico, Università di Salerno, 84100 Salerno (Italy)

The research reported in this document 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 n. DAJA-37-81-C-0740.

b) parameter values: none
c) dimension: N = 2
d) region: D = { $|x| \leq 2$, $|y| \leq 2$ }
e) penalization: none
f) minima in the region D : a global minimum at
   (x,y) = (0,-1), where f(x,y) = 3
   and three other local minima, at
   (x,y) = (-0.6, -0.4), where f = 30
   (x,y) = (-1.8, 2), where f = 84
   (x,y) = (-1.2, 0.8), where f = 840
g) initial point: $(x_o, y_o) = (1, 1)$
h) source: Appendix 1 in ref. [3]
i) notes: none.

Problem 17
Penalized Branin function

a) function:

$$f(x,y) = (y - bx^2 + cx - 6)^2 + 10\,[(1 - f)\cos x + 1]$$

b) parameter values: $b = 5.1/(4\pi^2)$, $c = 5/\pi$, $f = 1/(8\pi)$
c) dimension: N = 2
d) region: D = { $-5 \leq x \leq 10$, $0 \leq y \leq 15$ }
e) penalization:
   $w(x,y) = u(x - 2.5,7.5,100,2) + u(y - 7.5,7.5,100,2)$
f) minima in the region D : three global minima, at
   $(x,y) = (-\pi, 12.275)$
   $(x,y) = (\pi, 2.275)$
   $(x,y) = (3\pi, 2.475)$, where f = 0.39789
g) initial point: $(x_o, y_o) = (2.5, 7.5)$
h) source: Appendix 1 in ref. [3]
i) notes: the function has no non-global minima.

Problem 18
Penalized Shekel function, M = 5

a) function:

$$f(x) = -\sum_{i=1}^{M} \frac{1}{\sum_{j=1}^{N} [(x_j - a_{ij})^2 + c_i]}$$

where

$$c = [c_i] = [0.1, 0.2, 0.2, 0.4, 0.4, 0.6, 0.3, 0.7, 0.5, 0.5]^T$$

and

$$A = [a_{ij}] = \begin{vmatrix} 4 & 4 & 4 & 4 \\ 1 & 1 & 1 & 1 \\ 8 & 8 & 8 & 8 \\ 6 & 6 & 6 & 6 \\ 3 & 7 & 3 & 7 \\ 2 & 9 & 2 & 9 \\ 5 & 5 & 3 & 3 \\ 8 & 1 & 8 & 1 \\ 6 & 2 & 6 & 2 \\ 7 & 3.6 & 7 & 3.6 \end{vmatrix}$$

b) parameter values: M = 5
c) dimension: N = 4
d) region: $D = \{ 0 \leq x_j \leq 10, \quad j = 1, \ldots, N \}$
e) penalization:

$$w(x) = \sum_{j=1}^{N} u(x_j - 5, 5, 100, 2)$$

f) minima in the region D : a global minimum at

x = (4.00004, 4.00013, 4.00004, 4.00013), where f = -10.15320
and four other local minima, at
x = (1.00013, 1.00016, 1.00013, 1.00016), where f = -5.05520
x = (3.00180, 6.99833, 3.00180, 6.99833), where f = -2.63047
x = (5.99875, 6.00029, 5.99875, 6.00029), where f = -2.68286
x = (7.99958, 7.99964, 7.99958, 7.99964), where f = -5.10077

g) initial point: $x_{oj} = 9, \quad j = 1,\ldots,N$

h) source: Appendix 1 in ref. [3]

i) notes: the vectors of the coordinates of the M minimizers are very close to the first M row-vectors of the matrix A .

Problem 19
Penalized Shekel function, M = 7

a) function: as in problem 18
b) parameter values: M = 7
c), d), e) : as in problem 18
f) minima in the region D : a global minimum at

x = (4.00057, 4.00069, 3.99949, 3.99961), where f = -10.40294
and six other local minima, at
x = (1.00023, 1.00027, 1.00018, 1.00022), where f = -5.08767
x = (2.00481, 8.99168, 2.00462, 9.99150), where f = -1.83759
x = (3.00091, 7.00064, 3.00037, 7.00010), where f = -2.76590
x = (4.99423, 4.99500, 3.00606, 3.00683), where f = -3.72430
x = (5.99811, 6.00008, 5.99733, 5.99931), where f = -2.75193
x = (7.99951, 7.99962, 7.99950, 7.99961), where f = -5.10077

g), h), i) : as in problem 18.

Problem 20
Penalized Shekel function, M = 10

a) function: as in problem 18
b) parameter values: M = 10
c), d), e) : as in problem 18
f) minima in the region D : a global minimum at

x = (4.00075, 4.00059, 3.99966, 3.99951), where f = -10.53641
and nine other local minima, at
x = (1.00037, 1.00030, 1.00032, 1.00032), where f = -5.12848
x = (2.00510, 8.99129, 2.00491, 8.99111), where f = -1.85948
x = (3.00127, 7.00023, 3.00073, 6.99969), where f = -2.80663
x = (4.99487, 4.99398, 3.00756, 3.00667), where f = -3.83543
x = (5.99901, 5.99728, 5.99824, 5.99651), where f = -2.87114
x = (6.00558, 2.01001, 6.00437, 2.00881), where f = -2.42173
x = (6.99164, 3.59558, 6.99066, 3.59460), where f = -2.42734
x = (7.98678, 1.01224, 7.98644, 1.01190), where f = -1.67655
x = (7.99948, 7.99945, 7.99946, 7.99944), where f = -5.17565

g), h), i) : as in problem 18.

Problem 21
Penalized three-dimensional Hartman function, N = 3

a) function:

$$f(x) = - \sum_{i=1}^{M} c_i \exp \left[- \sum_{j=1}^{N} a_{ij} (x_j - p_{ij})^2 \right]$$

where

$$c = [c_i] = [\ 1,\ 1.2,\ 3,\ 3.2\ ]^T$$

b) parameter values: M = 4

$$A = [\, a_{ij} \,] = \begin{vmatrix} 3 & 10 & 30 \\ 0.1 & 10 & 35 \\ 3 & 10 & 30 \\ 0.1 & 10 & 35 \end{vmatrix}$$

$$P = [\, p_{ij} \,] = \begin{vmatrix} 0.3689 & 0.1170 & 0.2673 \\ 0.4699 & 0.4387 & 0.7470 \\ 0.1091 & 0.8732 & 0.5547 \\ 0.03815 & 0.5743 & 0.8828 \end{vmatrix}$$

c) dimension: N = 3

d) region: $D = \{\, 0 \leq x_j \leq 1, \quad j = 1,\ldots,N \,\}$

e) penalization:

$$w(x) = \sum_{j=1}^{N} u(x_j - 0.5, 0.5, 100, 2)$$

f) minima in the region D : a global minimum at

x = (0.11461, 0.55565, 0.85255), where f = -3.86278
and two other local minima at
x = (0.10934, 0.86052, 0.56412), where f = -3.08976
x = (0.36872, 0.11756, 0.26757), where f = -1.00082

g) initial point: $x_{oj} = 0.5, \quad j = 1,\ldots,N$

h) source: Appendix 1 in ref. [3]

i) notes: none.

Problem 22
Penalized six-dimensional Hartman function, N = 6

a) function: as in problem 21
b) parameter values: M = 4

$$A = [\,a_{ij}\,] = \begin{vmatrix} 10 & 3 & 17 & 3.5 & 1.7 & 8 \\ .05 & 10 & 17 & .1 & 8 & 14 \\ 3 & 3.5 & 1.7 & 10 & 17 & 8 \\ 17 & 8 & .05 & 10 & 0.1 & 14 \end{vmatrix}$$

$$P = [\,p_{ij}\,] = \begin{vmatrix} 0.1312 & 0.1696 & 0.5569 & 0.0124 & 0.8283 & 0.5886 \\ 0.2329 & 0.4135 & 0.8307 & 0.3736 & 0.1004 & 0.9991 \\ 0.2348 & 0.1451 & 0.3522 & 0.2883 & 0.3047 & 0.6650 \\ 0.4047 & 0.8828 & 0.8732 & 0.5743 & 0.1091 & 0.0381 \end{vmatrix}$$

c) dimension: N = 6

d) region: $D = \{\, 0 \le x_j \le 1, \quad j = 1,\ldots,N \,\}$

e) penalization:

$$w(x) = \sum_{j=1}^{N} u(x_j - 0.5, 0.5, 100, 2)$$

f) minima in the region D : a global minimum at

x = (0.20169, 0.15001, 0.47687, 0.27533, 0.31165, 0.65730)
where f = -3.32237, and another local minimum at
x = (0.40465, 0.88244, 0.84610, 0.57399, 0.13893, 0.038496)
where f = -3.20316

g) initial point: $x_{oj} = 0.5, \quad j = 1,\ldots,N$

h) source: Appendix 1 in ref. [3]
i) notes: none.

Problem 23
Penalized Levy-Montalvo function, type 1, N = 2

a) function:

$$f(x) = (\pi/N)\ \{\ 10\ \sin^2(\pi y_1) + (y_N - 1)^2 + \sum_{j=1}^{N-1} (y_j - 1)^2\ [1 + 10\ \sin^2(\pi y_{j+1})]\ \}$$

where $y_j = 1 + (x_j - 1)/4, \quad j = 1,\ldots,N$

b) parameter values: none
c) dimension: N = 2
d) region: $D = \{\ -10 \le x_j \le 10, \quad j = 1,\ldots,N\ \}$
e) penalization:

$$w(x) = \sum_{j=1}^{N} u(x_j,10,100,4)$$

f) minima in the region D : a single global minimum at
$x_j = 1, \quad j = 1,\ldots,N$, where $f = 0$
and a number of local minima of the order of 5^N
g) initial point: $x_{oj} = 0, \quad j = 1,\ldots,N$
h) source: ref. [5]
i) notes: three-dimensional plots of f are given in [5].

Problem 24
Penalized Levy-Montalvo function, type 1, N = 3

a), b) : as in problem 23
c) dimension: N = 3
d), e), f), g), h), i) : as in problem 23.

Problem 25
Penalized Levy-Montalvo function, type 1, N = 4

a), b) : as in problem 23
c) dimension: N = 4
d), e), f), g), h), i) : as in problem 23.

Problem 26
Penalized Levy-Montalvo function, type 2, N = 5

a) function: as in problem 23, but with

$$y_i = x_i , \quad i = 1,\ldots,N$$

b) : as in problem 23
c) dimension: N = 5
d), e) : as in problem 23
f) minima in the region D :
a single global minimum as in problem 23,
and a number of local minima of the order of 10^N
g), h), i) : as in problem 23.

Problem 27
Penalized Levy-Montalvo function, type 2, N = 8

a), b) : as in problem 26
c) dimension: N = 8
d), e), f), g), h), i) : as in problem 26.

Problem 28
Penalized Levy-Montalvo function, type 2, N = 10

a), b) : as in problem 26
c) dimension: N = 10
d), e), f), g), h), i) : as in problem 26.

Problem 29
Penalized Levy-Montalvo function, type 3, range 10, N = 2

a) function:

$$f(x) = 0.1 \{ \sin^2(3\pi x_1) + (x_N - 1)^2 [1 + \sin^2(2\pi x_N)] + \sum_{i=1}^{N-1} (x_i - 1)^2 [1 + \sin^2(3\pi x_{i+1})] \}$$

b) parameter values: none
c) dimension: N = 2
d), e) : as in problem 23
f) minima in the region D : a single global minimum at

$x_j = 1, \quad j = 1,\ldots,N$, where $f = 0$

and a number of non-global minima of the order of 30^N

g), h), i) : as in probl. 23.

Problem 30
Penalized Levy-Montalvo function, type 3, range 10, N = 3

a), b) : as in problem 29
c) dimension: N = 3
d), e), f), g), h), i) : as in problem 29.

Problem 31
Penalized Levy-Montalvo function, type 3, range 10, N = 4

a), b) : as in problem 29
c) dimension: N = 4
d), e), f), g), h), i) : as in problem 29.

Problem 32
Penalized Levy-Montalvo function, type 3, range 5, N = 5

a), b) : as in problem 29
c) dimension: N = 5
d) region:

d) region: $D = \{ -5 \le x_i \le 5, \quad i = 1,\ldots,N \}$

e) penalization:

$$w(x) = \sum_{i=1}^{N} u(x_i,5,100,4)$$

f) minima in the region D :

a single global minimum as in problem 29,

and a number of local minima of the order of 15^N

g), h), i) : as in problem 29.

Problem 33
Penalized Levy-Montalvo function, type 3, range 5, N = 6

a), b) : as in problem 32
c) dimension: N = 6
d), e), f), g), h), i) : as in problem 32.

Problem 34
Penalized Levy-Montalvo function, type 3, range 5, N = 7

a), b) : as in problem 32
c) dimension: N = 7
d), e), f), g), h), i) : as in problem 32.

Problem 35
A function with a cusp-shaped minimum

a) function:

$$f(x) = \left(\sum_{j=1}^{5} j\, x_j^2 \right)^{1/4}$$

b) parameter values: none
c) dimension: N = 5
d) region: $D = \{ -20000 \le x_j \le 10000, \quad j = 1,\ldots,N \}$
e) penalization: none
f) minima in the region D : a single global minimum at

$x_j = 0, \quad j = 1,\ldots,N$, where $f = 0$

g) initial point: $x_{oj} = 1000, \quad j = 1,\ldots,N$

h) source: suggested by one of the authors (V.P.)
i) notes: non-differentiable problem: the only (global) minimizer is a singular point with unbounded derivative; the eigenvalues of the hessian matrix are everywhere of mixed sign.

Problem 36
A function with a global minimum having a small region of attraction, a = 10, N = 2

a) function:

$$f(x) = \|x\|^2 - (C + h)\, g(x)$$

where

$$g(x) = \exp\left(- S / \left(b^2 - S \right) \right) \qquad (S < b^2)$$
$$= 0 \qquad (S \geq b^2)$$

$$S = \|x - c\|^2, \qquad C = \|c\|^2$$

and

$$x = (x_1, x_2, \ldots, x_N), \qquad c = (a, 0, 0, \ldots, 0)$$

b) parameter values: a = 100, b = 1, h = 10
c) dimension: N = 2
d) region:

$$D = \{ -1000 \leq x_j \leq 1000, \quad j = 1, \ldots, N \}$$

e) penalization: none
f) minima in the region D : a single global minimum at
x = (99.99001, 0), where f = -10.99885
and another local minimum at
x = (0, 0), where f = 0

g) initial point: $x_0 = (0, 100)$

h) source: suggested by one of the authors (F.A.P.)

i) notes: the perturbation to $\|x\|^2$, which contains the global minimum, is "visible" only in a small neighborhood of the point c .

Problem 37
A function with a global minimum having a small region of attraction, a = 100, N = 5

a) function: as in problem 36
b) parameter values: a = 10, b = 1, h = 10
c) dimension: N = 5
d) region: $D = \{ -100 \leq x_j \leq 100, \quad j = 1,\ldots,N \}$
e) penalization: none
f) minima in the region D : a single global minimum at
x = (9.91062, 0, 0, 0, 0), where f = -10.89732
and another local minimum at
x = (0, 0, 0, 0, 0), where f = 0
g) initial point: x_0 = (0, 0, 0, 0, 10)
h), i) : as in probl. 36.

APPENDIX 2

The FORTRAN listing

```
      N = 5                                                              GLIP2860
      V0 = 1000.D0                                                       GLIP2870
      VMIN = -20000.D0                                                   GLIP2880
      VMAX = 10000.D0                                                    GLIP2890
      GO TO 800                                                          GLIP2900
C                                                                        GLIP2910
C   36  A FUNCTION WITH A SMALL-ATTRACTION-REGION GLOBAL MINIMUM (N = 2)GLIP2920
C                                                                        GLIP2930
  360 CONTINUE                                                           GLIP2940
      N = 2                                                              GLIP2950
      X0(1) = 0.D0                                                       GLIP2960
      X0(2) = 100.D0                                                     GLIP2970
      VMIN = -1000.D0                                                    GLIP2980
      VMAX = 1000.D0                                                     GLIP2990
      GO TO 700                                                          GLIP3000
C                                                                        GLIP3010
C   37  A FUNCTION WITH A SMALL-ATTRACTION-REGION GLOBAL MINIMUM (N = 5)GLIP3020
C                                                                        GLIP3030
  370 CONTINUE                                                           GLIP3040
      N = 5                                                              GLIP3050
      X0(1) = 0.D0                                                       GLIP3060
      X0(2) = 0.D0                                                       GLIP3070
      X0(3) = 0.D0                                                       GLIP3080
      X0(4) = 0.D0                                                       GLIP3090
      X0(5) = 10.D0                                                      GLIP3100
      VMIN = -100.D0                                                     GLIP3110
      VMAX = 100.D0                                                      GLIP3120
      GO TO 700                                                          GLIP3130
C                                                                        GLIP3140
  700 CONTINUE                                                           GLIP3150
      DO 710 I = 1,N                                                     GLIP3160
         XMIN(I) = VMIN                                                  GLIP3170
         XMAX(I) = VMAX                                                  GLIP3180
  710 CONTINUE                                                           GLIP3190
      RETURN                                                             GLIP3200
C                                                                        GLIP3210
  800 CONTINUE                                                           GLIP3220
      DO 810 I = 1,N                                                     GLIP3230
         XMIN(I) = VMIN                                                  GLIP3240
         XMAX(I) = VMAX                                                  GLIP3250
  810 CONTINUE                                                           GLIP3260
C                                                                        GLIP3270
  900 CONTINUE                                                           GLIP3280
      DO 910 I = 1,N                                                     GLIP3290
         X0(I) = V0                                                      GLIP3300
  910 CONTINUE                                                           GLIP3310
      RETURN                                                             GLIP3320
      END                                                                GLIP3330
```

```
      GO TO 315                                                          GLIP2290
C                                                                        GLIP2300
C   28  PENALIZED LEVY-MONTALVO FUNCTION, TYPE 2  (N = 10)               GLIP2310
C                                                                        GLIP2320
  280 CONTINUE                                                           GLIP2330
      N = 10                                                             GLIP2340
      GO TO 315                                                          GLIP2350
C                                                                        GLIP2360
C   29  PENALIZED LEVY-MONTALVO FUNCTION, TYPE 3, RANGE = 10   (N = 2)  GLIP2370
C                                                                        GLIP2380
  290 CONTINUE                                                           GLIP2390
      N = 2                                                              GLIP2400
      GO TO 315                                                          GLIP2410
C                                                                        GLIP2420
C   30  PENALIZED LEVY-MONTALVO FUNCTION, TYPE 3, RANGE = 10   (N = 3)  GLIP2430
C                                                                        GLIP2440
  300 CONTINUE                                                           GLIP2450
      N = 3                                                              GLIP2460
      GO TO 315                                                          GLIP2470
C                                                                        GLIP2480
C   31  PENALIZED LEVY-MONTALVO FUNCTION, TYPE 3, RANGE = 10   (N = 4)  GLIP2490
C                                                                        GLIP2500
  310 CONTINUE                                                           GLIP2510
      N = 4                                                              GLIP2520
C                                                                        GLIP2530
  315 CONTINUE                                                           GLIP2540
      V0 = 0.00                                                          GLIP2550
      VMIN = -10.00                                                      GLIP2560
      VMAX = 10.00                                                       GLIP2570
      GO TO 900                                                          GLIP2580
C                                                                        GLIP2590
C   32  PENALIZED LEVY-MONTALVO FUNCTION, TYPE 3, RANGE = 5   (N = 5)   GLIP2600
C                                                                        GLIP2610
  320 CONTINUE                                                           GLIP2620
      N = 5                                                              GLIP2630
      GO TO 345                                                          GLIP2640
C                                                                        GLIP2650
C   33  PENALIZED LEVY-MONTALVO FUNCTION, TYPE 3, RANGE = 5   (N = 6)   GLIP2660
C                                                                        GLIP2670
  330 CONTINUE                                                           GLIP2680
      N = 6                                                              GLIP2690
      GO TO 345                                                          GLIP2700
C                                                                        GLIP2710
C   34  PENALIZED LEVY-MONTALVO FUNCTION, TYPE 3, RANGE = 5  (N = 7)    GLIP2720
C                                                                        GLIP2730
  340 CONTINUE                                                           GLIP2740
      N = 7                                                              GLIP2750
C                                                                        GLIP2760
  345 CONTINUE                                                           GLIP2770
      V0 = 0.00                                                          GLIP2780
      VMIN = -5.00                                                       GLIP2790
      VMAX = 5.00                                                        GLIP2800
      GO TO 900                                                          GLIP2810
C                                                                        GLIP2820
C   35  A FUNCTION WITH A SINGLE CUSP-SHAPED MINIMUM  (N = 5)            GLIP2830
C                                                                        GLIP2840
  350 CONTINUE                                                           GLIP2850
```

```
C   19  PENALIZED SHEKEL FUNCTION, M = 7  (N = 4)                    GLIP1720
C                                                                    GLIP1730
  190 CONTINUE                                                       GLIP1740
C                                                                    GLIP1750
C   20  PENALIZED SHEKEL FUNCTION, M = 10  (N = 4)                   GLIP1760
C                                                                    GLIP1770
  200 CONTINUE                                                       GLIP1780
      N = 4                                                          GLIP1790
      VC = 9.D0                                                      GLIP1800
      VMIN = 0.D0                                                    GLIP1810
      VMAX = 10.D0                                                   GLIP1820
      GO TO 800                                                      GLIP1830
C                                                                    GLIP1840
C   21  PENALIZED THREE-DIMENSIONAL HARTMAN FUNCTION  (N = 3)        GLIP1850
C                                                                    GLIP1860
  210 CONTINUE                                                       GLIP1870
      N = 3                                                          GLIP1880
      GO TO 225                                                      GLIP1890
C                                                                    GLIP1900
C   22  PENALIZED SIX-DIMENSIONAL HARTMAN FUNCTION  (N = 6)          GLIP1910
C                                                                    GLIP1920
  220 CONTINUE                                                       GLIP1930
      N = 6                                                          GLIP1940
  225 CONTINUE                                                       GLIP1950
      VO = 0.5D0                                                     GLIP1960
      VMIN = 0.D0                                                    GLIP1970
      VMAX = 1.D0                                                    GLIP1980
      GO TO 800                                                      GLIP1990
C                                                                    GLIP2000
C   23  PENALIZED LEVY-MONTALVO FUNCTION, TYPE 1  (N = 2)            GLIP2010
C                                                                    GLIP2020
  230 CONTINUE                                                       GLIP2030
      N = 2                                                          GLIP2040
      GO TO 315                                                      GLIP2050
C                                                                    GLIP2060
C   24  PENALIZED LEVY-MONTALVO FUNCTION, TYPE 1  (N = 3)            GLIP2070
C                                                                    GLIP2080
  240 CONTINUE                                                       GLIP2090
      N = 3                                                          GLIP2100
      GO TO 315                                                      GLIP2110
C                                                                    GLIP2120
C   25  PENALIZED LEVY-MONTALVO FUNCTION, TYPE 1  (N = 4)            GLIP2130
C                                                                    GLIP2140
  250 CONTINUE                                                       GLIP2150
      N = 4                                                          GLIP2160
      GO TO 315                                                      GLIP2170
C                                                                    GLIP2180
C   26  PENALIZED LEVY-MONTALVO FUNCTION, TYPE 1  (N = 5)            GLIP2190
C                                                                    GLIP2200
  260 CONTINUE                                                       GLIP2210
      N = 5                                                          GLIP2220
      GO TO 315                                                      GLIP2230
C                                                                    GLIP2240
C   27  PENALIZED LEVY-MONTALVO FUNCTION, TYPE 2  (N = 8)            GLIP2250
C                                                                    GLIP2260
  270 CONTINUE                                                       GLIP2270
      N = 8                                                          GLIP2280
```

```
C    12  A FUNCTION WITH THREE ILL-CONDITIONED MINIMA, A=10  (N = 2)      GLIP1150
C                                                                         GLIP1160
  120 CONTINUE                                                            GLIP1170
C                                                                         GLIP1180
C    11  A FUNCTION WITH THREE ILL-CONDITIONED MINIMA, A=10**2  (N = 2)   GLIP1190
C                                                                         GLIP1200
  110 CONTINUE                                                            GLIP1210
C                                                                         GLIP1220
C    14  A FUNCTION WITH THREE ILL-CONDITIONED MINIMA, A=10**3  (N = 2)   GLIP1230
C                                                                         GLIP1240
  140 CONTINUE                                                            GLIP1250
C                                                                         GLIP1260
C    13  A FUNCTION WITH THREE ILL-CONDITIONED MINIMA, A=10**4  (N = 2)   GLIP1270
C                                                                         GLIP1280
  130 CONTINUE                                                            GLIP1290
C                                                                         GLIP1300
C    16  A FUNCTION WITH THREE ILL-CONDITIONED MINIMA, A=10**5  (N = 2)   GLIP1310
C                                                                         GLIP1320
  160 CONTINUE                                                            GLIP1330
C                                                                         GLIP1340
C    15  A FUNCTION WITH THREE ILL-CONDITIONED MINIMA, A=10**6  (N = 2)   GLIP1350
C                                                                         GLIP1360
  150 CONTINUE                                                            GLIP1370
      N = 2                                                               GLIP1380
      XO(1) = 0.D0                                                        GLIP1390
      XO(2) = 0.D0                                                        GLIP1400
      XMIN(1) = -10.D0                                                    GLIP1410
      XMAX(1) = 10.D0                                                     GLIP1420
      XMIN(2) = -100.D0                                                   GLIP1430
      XMAX(2) = 100.D0                                                    GLIP1440
      RETURN                                                              GLIP1450
C                                                                         GLIP1460
C    18  GOLDSTEIN-PRICE FUNCTION  (N = 2)                                GLIP1470
C                                                                         GLIP1480
  180 CONTINUE                                                            GLIP1490
      N = 2                                                               GLIP1500
      VO = 1.D0                                                           GLIP1510
      VMIN = -2.D0                                                        GLIP1520
      VMAX = 2.D0                                                         GLIP1530
      GO TO 900                                                           GLIP1540
C                                                                         GLIP1550
C    17  PENALIZED BRANIN FUNCTION  (N = 2)                               GLIP1560
C                                                                         GLIP1570
  170 CONTINUE                                                            GLIP1580
      N = 2                                                               GLIP1590
      XO(1) = 2.5D0                                                       GLIP1600
      XO(2) = 7.5D0                                                       GLIP1610
      XMIN(1) = -5.D0                                                     GLIP1620
      XMAX(1) = 10.D0                                                     GLIP1630
      XMIN(2) = 0.D0                                                      GLIP1640
      XMAX(2) = 15.D0                                                     GLIP1650
      RETURN                                                              GLIP1660
C                                                                         GLIP1670
C    1   PENALIZED SHEKEL FUNCTION, M = 5  (N = 4)                       GLIP1680
C                                                                         GLIP1690
  1 CONTINUE                                                              GLIP1700
                                                                          GLIP1710
```

```
   30 CONTINUE                                                          GLIP0580
      N = 1                                                             GLIP0590
      V0 = 0.D0                                                         GLIP0600
      VMIN = -10.D0                                                     GLIP0610
      VMAX = 10.D0                                                      GLIP0620
      GOTO 900                                                          GLIP0630
C                                                                       GLIP0640
C     4  A FOURTH ORDER POLYNOMIAL IN TWO VARIABLES  (N = 2)            GLIP0650
C                                                                       GLIP0660
   40 CONTINUE                                                          GLIP0670
      N = 2                                                             GLIP0680
      X0(1) = 1.D0                                                      GLIP0690
      X0(2) = 0.D0                                                      GLIP0700
      VMIN = -10.D0                                                     GLIP0710
      VMAX = 10.D0                                                      GLIP0720
      GO TO 900                                                         GLIP0730
C                                                                       GLIP0740
C     5  A FUNCTION WITH A SINGLE ROW OF LOCAL MINIMA  (N = 2)          GLIP0750
C                                                                       GLIP0760
   50 CONTINUE                                                          GLIP0770
      N = 2                                                             GLIP0780
      X0(1) = -3.D0                                                     GLIP0790
      X0(2) = 0.D0                                                      GLIP0800
      XMIN(1) = -15.D0                                                  GLIP0810
      XMAX(1) = 25.D0                                                   GLIP0820
      XMIN(2) = -5.D0                                                   GLIP0830
      XMAX(2) = 15.D0                                                   GLIP0840
      RETURN                                                            GLIP0850
C                                                                       GLIP0860
C     6  SIX-HUMP CAMEL FUNCTION  (N = 2)                               GLIP0870
C                                                                       GLIP0880
   60 CONTINUE                                                          GLIP0890
      N = 2                                                             GLIP0900
      V0 = 0.D0                                                         GLIP0910
      XMIN(1) = -3.D0                                                   GLIP0920
      XMAX(1) = 3.D0                                                    GLIP0930
      XMIN(2) = -2.D0                                                   GLIP0940
      XMAX(2) = 2.D0                                                    GLIP0950
      GO TO 900                                                         GLIP0960
C                                                                       GLIP0970
C     7  TWO-DIMENSIONAL PENALIZED SHUBERT FUNCTION, BETA = 0  (N = 2)  GLIP0980
C                                                                       GLIP0990
   70 CONTINUE                                                          GLIP1000
C                                                                       GLIP1010
C     8  TWO-DIMENSIONAL PENALIZED SHUBERT FUNCTION, BETA = 1/2  (N = 2) GLIP1020
C                                                                       GLIP1030
   80 CONTINUE                                                          GLIP1040
C                                                                       GLIP1050
C     9  TWO-DIMENSIONAL PENALIZED SHUBERT FUNCTION, BETA = 1  (N = 2)  GLIP1060
C                                                                       GLIP1070
   90 CONTINUE                                                          GLIP1080
      N = 2                                                             GLIP1090
      V0 = 0.D0                                                         GLIP1100
      VMIN = -10.D0                                                     GLIP1110
      VMAX = 10.D0                                                      GLIP1120
      GO TO 900                                                         GLIP1130
                                                                        GLIP1140
```

```
      SUBROUTINE GLOMIP (NPROB, N, XO, XMIN, XMAX)                      GLIP0010
C                                                                       GLIP0020
C     THE SUBROUTINE  GLOMIP  PROVIDES THE CODING FOR THE NUMBER        GLIP0030
C     OF VARIABLES, THE INITIAL POINT, AND THE OBSERVATION REGION       GLIP0040
C     TO BE USED, TOGETHER WITH THE 37 TEST FUNCTIONS GIVEN BY          GLIP0050
C     SUBROUTINE  GLOMTF , TO DEFINE 37 TEST PROBLEMS FOR GLOBAL        GLIP0060
C     MINIMIZATION SOFTWARE.                                            GLIP0070
C                                                                       GLIP0080
C     THE SUBROUTINE  GLOMIP  RETURNS IN  N , XO , AND  XMIN ,          GLIP0090
C     XMAX  THE NUMBER OF VARIABLES, THE INITIAL POINT, AND THE         GLIP0100
C     BOUNDARIES OF THE OBSERVATION REGION.                             GLIP0110
C                                                                       GLIP0120
C     CALLING STATMENT                                                  GLIP0130
C        CALL GLOMIP (NPROB, N, XO, XMIN, XMAX)                         GLIP0140
C                                                                       GLIP0150
C     DESCRIPTION OF THE CALL PARAMETERS                                GLIP0160
C        (THE FORTRAN IMPLICIT TYPE DEFINITION FOR INTEGERS IS USED.    GLIP0170
C        ALL NON-INTEGER PARAMETERS ARE DOUBLE-PRECISION).              GLIP0180
C                                                                       GLIP0190
C     NPROB    IS THE (INPUT) NUMBER OF THE TEST PROBLEM TO BE          GLIP0200
C              CONSIDERED                                               GLIP0210
C     N        IS THE (OUTPUT) NUMBER OF VARIABLES (DIMENSION) OF       GLIP0220
C              THE PROBLEM                                              GLIP0230
C     XMIN , XMAX   ARE THE (OUTPUT) N-VECTORS CONTAINING THE LEFT      GLIP0240
C              AND RIGHT BOUNDARIES OF THE OBSERVATION REGION           GLIP0250
C              DEFINED BY THE POINTS  X = (X1,...,XN)  SUCH THAT        GLIP0260
C                 XMIN(I) .LE. X(I) .LE. XMAX(I),  I = 1,...,N          GLIP0270
C                                                                       GLIP0280
      DOUBLE PRECISION XO,XMIN,XMAX                                     GLIP0290
      DOUBLE PRECISION VO,VMIN,VMAX                                     GLIP0300
C                                                                       GLIP0310
      DIMENSION XO(1),XMIN(1),XMAX(1)                                   GLIP0320
C                                                                       GLIP0330
      GO TO (10,20,30,40,50,60,70,80,90,100,110,120,130,140,150,        GLIP0340
     1       160,170,180,190,200,210,220,230,240,250,260,270,280,       GLIP0350
     2       290,300,310,320,330,340,350,360,370), NPROB                GLIP0360
C                                                                       GLIP0370
C     1  A FOURTH-ORDER POLYNOMIAL  (N = 1)                             GLIP0380
C                                                                       GLIP0390
   10 CONTINUE                                                          GLIP0400
      N = 1                                                             GLIP0410
      VO = 1.D0                                                         GLIP0420
      VMIN = -10.D0                                                     GLIP0430
      VMAX = 10.D0                                                      GLIP0440
      GO TO 800                                                         GLIP0450
C                                                                       GLIP0460
C     2  GOLDSTEIN SIXTH ORDER POLYNOMIAL  (N = 1)                      GLIP0470
C                                                                       GLIP0480
   20 CONTINUE                                                          GLIP0490
      N = 1                                                             GLIP0500
      VO = 0.D0                                                         GLIP0510
      VMIN = -4.D0                                                      GLIP0520
      VMAX = 4.D0                                                       GLIP0530
      GOTO 800                                                          GLIP0540
C                                                                       GLIP0550
C     3  ONE-DIMENSIONAL PENALIZED SHUBERT FUNCTION  (N = 1)            GLIP0560
C                                                                       GLIP0570
```

```
      DO 3?5 I = 1,N                                                    GLTF4000
         FUNZ = FUNZ+DBLE(FLOAT(I))*X(I)*X(I)                           GLTF4010
  3?5 CONTINUE                                                          GLTF4020
      FUNZ = DSQRT(DSQRT(FUNZ))                                         GLTF4030
      RETURN                                                            GLTF4040
C                                                                       GLTF4050
C   ?6  A FUNCTION WITH A SMALL-ATTRACTION-REGION GLOBAL MINIMUM (N = 2)GLTF4060
C                                                                       GLTF4070
  360 CONTINUE                                                          GLTF4080
      RG = P?6A                                                         GLTF4090
      RP = P?6B                                                         GLTF4100
      H = P?6C                                                          GLTF4110
      PUND = P?6D                                                       GLTF4120
      GOTO 373                                                          GLTF4130
C                                                                       GLTF4140
C   ?7  A FUNCTION WITH A SMALL-ATTRACTION-REGION GLOBAL MINIMUM (N = 5)GLTF4150
C                                                                       GLTF4160
  370 CONTINUE                                                          GLTF4170
      RG = P?7A                                                         GLTF4180
      RP = P?7B                                                         GLTF4190
      H = P?7C                                                          GLTF4200
      PUND = P?7D                                                       GLTF4210
  373 CONTINUE                                                          GLTF4220
      S = 0.D0                                                          GLTF4230
      DO 377 I = 2,N                                                    GLTF4240
         S = S+X(I)*X(I)                                                GLTF4250
  377 CONTINUE                                                          GLTF4260
      FUNZ = S+X(1)*X(1)                                                GLTF4270
      S = S+(X(1)-RG)**2                                                GLTF4280
      IF (S.LT.RP*RP*PUND) FUNZ = FUNZ-(RG*RG+H)*DEXP(-S/(RP*RP-S))     GLTF4290
      RETURN                                                            GLTF4300
C                                                                       GLTF4310
      END                                                               GLTF4320
```

```
      DO 295 I = 1,N                                                    GLTF3430
         Y(I) = X(I)                                                    GLTF3440
  295 CONTINUE                                                          GLTF3450
C                                                                       GLTF3460
  296 CONTINUE                                                          GLTF3470
      FUNZ = 10.D0*DSIN(PI*Y(1))**2+(Y(N)-1.D0)**2                      GLTF3480
      DO 297 I=2,N                                                      GLTF3490
         FUNZ = FUNZ+(Y(I-1)-1.D0)**2*(1.D0+10.D0*DSIN(PI*Y(I))**2)     GLTF3500
  297 CONTINUE                                                          GLTF3510
      FUNZ = FUNZ*PI/DBLE(FLOAT(N))                                     GLTF3520
      RANGE = 10.D0                                                     GLTF3530
      GO TO 347                                                         GLTF3540
C                                                                       GLTF3550
C   30  PENALIZED LEVY-MONTALVO FUNCTION, TYPE 3, RANGE = 10    (N = 2)  GLTF3560
C                                                                       GLTF3570
  298 CONTINUE                                                          GLTF3580
C                                                                       GLTF3590
C   30  PENALIZED LEVY-MONTALVO FUNCTION, TYPE 3, RANGE = 10    (N = 3)  GLTF3600
C                                                                       GLTF3610
  300 CONTINUE                                                          GLTF3620
C                                                                       GLTF3630
C   31  PENALIZED LEVY-MONTALVO FUNCTION, TYPE 3, RANGE = 10    (N = 4)  GLTF3640
C                                                                       GLTF3650
  310 CONTINUE                                                          GLTF3660
      RANGE = 10.D0                                                     GLTF3670
      GO TO 342                                                         GLTF3680
C                                                                       GLTF3690
C   32  PENALIZED LEVY-MONTALVO FUNCTION, TYPE 3, RANGE = 5     (N = 5)  GLTF3700
C                                                                       GLTF3710
  320 CONTINUE                                                          GLTF3720
C                                                                       GLTF3730
C   33  PENALIZED LEVY-MONTALVO FUNCTION, TYPE 3, RANGE = 5     (N = 6)  GLTF3740
C                                                                       GLTF3750
  330 CONTINUE                                                          GLTF3760
C                                                                       GLTF3770
C   34  PENALIZED LEVY-MONTALVO FUNCTION, TYPE 3, RANGE = 5     (N = 7)  GLTF3780
C                                                                       GLTF3790
  340 CONTINUE                                                          GLTF3800
      RANGE = 5.D0                                                      GLTF3810
  342 CONTINUE                                                          GLTF3820
      FUNZ = DSIN(3.D0*PI*X(1))**2                                      GLTF3830
     1   *(X(N)-1.D0)**2*(1.D0+DSIN(2.D0*PI*X(N))**2)                   GLTF3840
      DO 345 I=2,N                                                      GLTF3850
         FUNZ = FUNZ+(X(I-1)-1.D0)**2*(1.D0+DSIN(3.D0*PI*X(I))**2)      GLTF3860
  345 CONTINUE                                                          GLTF3870
      FUNZ = FUNZ*0.100                                                 GLTF3880
  347 CONTINUE                                                          GLTF3890
      DO 349 I = 1,N                                                    GLTF3900
         IF (DABS(X(I)).GT.RANGE)                                       GLTF3910
     1      FUNZ = FUNZ+PENFUN(X(I),RANGE,100.D0,4)                     GLTF3920
  349 CONTINUE                                                          GLTF3930
      RETURN                                                            GLTF3940
C                                                                       GLTF3950
C   35  A FUNCTION WITH A SINGLE CUSP-SHAPED MINIMUM  (N = 5)           GLTF3960
C                                                                       GLTF3970
  350 CONTINUE                                                          GLTF3980
      FUNZ = 0.0D0                                                      GLTF3990
```

```
      M = 4                                                              GLTF2860
      FUNZ = 0.D0                                                        GLTF2870
      DO 217 I = 1,M                                                     GLTF2880
         S = 0.D0                                                        GLTF2890
         DO 213 J = 1,N                                                  GLTF2900
            S = S-P21A(I,J)*(X(J)-P21B(I,J))**2                          GLTF2910
  213    CONTINUE                                                        GLTF2920
         IF (S.GE.P21D) FUNZ = FUNZ-P21C(I)*DEXP(S)                      GLTF2930
  217 CONTINUE                                                           GLTF2940
      GO TO 227                                                          GLTF2950
C                                                                        GLTF2960
C   22  PENALIZED SIX-DIMENSIONAL HARTMAN FUNCTION  (N = 6)              GLTF2970
C                                                                        GLTF2980
  220 CONTINUE                                                           GLTF2990
      M = 4                                                              GLTF3000
      FUNZ = 0.D0                                                        GLTF3010
      DO 225 I = 1,M                                                     GLTF3020
         S = 0.D0                                                        GLTF3030
         DO 223 J = 1,N                                                  GLTF3040
            S = S-P22A(I,J)*(X(J)-P22B(I,J))**2                          GLTF3050
  223    CONTINUE                                                        GLTF3060
         IF (S.GE.P22D) FUNZ = FUNZ-P22C(I)*DEXP(S)                      GLTF3070
  225 CONTINUE                                                           GLTF3080
  227 CONTINUE                                                           GLTF3090
      DO 229 I = 1,N                                                     GLTF3100
      IF (DABS(X(I)-0.500).GT.0.500)                                     GLTF3110
     1    FUNZ = FUNZ+PENFUN(X(I)-0.500,0.500,100.00,2)                  GLTF3120
  229 CONTINUE                                                           GLTF3130
      RETURN                                                             GLTF3140
C                                                                        GLTF3150
C   23  PENALIZED LEVY-MONTALVO FUNCTION, TYPE 1  (N = 2)                GLTF3160
C                                                                        GLTF3170
  230 CONTINUE                                                           GLTF3180
C                                                                        GLTF3190
C   24  PENALIZED LEVY-MONTALVO FUNCTION, TYPE 1  (N = 3)                GLTF3200
C                                                                        GLTF3210
  240 CONTINUE                                                           GLTF3220
C                                                                        GLTF3230
C   25  PENALIZED LEVY-MONTALVO FUNCTION, TYPE 1  (N = 4)                GLTF3240
C                                                                        GLTF3250
  250 CONTINUE                                                           GLTF3260
      DO 255 I = 1,N                                                     GLTF3270
         Y(I) = 1.D0+0.2500*(X(I)-1.00)                                  GLTF3280
  255 CONTINUE                                                           GLTF3290
      GO TO 265                                                          GLTF3300
C                                                                        GLTF3310
C   26  PENALIZED LEVY-MONTALVO FUNCTION, TYPE 2  (N = 5)                GLTF3320
C                                                                        GLTF3330
  260 CONTINUE                                                           GLTF3340
C                                                                        GLTF3350
C   27  PENALIZED LEVY-MONTALVO FUNCTION, TYPE 2  (N = 6)                GLTF3360
C                                                                        GLTF3370
  270 CONTINUE                                                           GLTF3380
C                                                                        GLTF3390
C   28  PENALIZED LEVY-MONTALVO FUNCTION, TYPE 2  (N = 10)               GLTF3400
C                                                                        GLTF3410
  280 CONTINUE                                                           GLTF3420
```

```
C   16  GOLDSTEIN-PRICE FUNCTION  (N = 2)                                  GLTF2290
C                                                                          GLTF2300
  160 CONTINUE                                                             GLTF2310
      U = X(1)+X(2)+1.D0                                                   GLTF2320
      V = 2.D0*X(1)-3.D0*X(2)                                              GLTF2330
      UU = U*U                                                             GLTF2340
      VV = V*V                                                             GLTF2350
      FUNZ = (1.D0+UU*(36.D0-2D.D0*U+3.D0*UU))                             GLTF2360
     1     * (30.D0+VV*(18.D0-16.D0*V+3.D0*VV))                            GLTF2370
      RETURN                                                               GLTF2380
C                                                                          GLTF2390
C   17  PENALIZED BRANIN FUNCTION  (N = 2)                                 GLTF2400
C                                                                          GLTF2410
  170 CONTINUE                                                             GLTF2420
      FUNZ = (X(2)-P17A*(X(1)/PI)**2+(5.D0/PI)*X(1)-6.D0)**2               GLTF2430
     1     +10.D0*(1.D0-1.D0/(8.D0*PI))*DCOS(X(1))+10.D0                   GLTF2440
      IF (DABS(X(1)-2.500).GT.7.5D0)                                       GLTF2450
     1   FUNZ = FUNZ+PENFUN(X(1)-2.500,7.500,100.D0,2)                     GLTF2460
      IF (DABS(X(2)-7.500).GT.7.500)                                       GLTF2470
     1   FUNZ = FUNZ+PENFUN(X(2)-7.500,7.500,100.D0,2)                     GLTF2480
      RETURN                                                               GLTF2490
C                                                                          GLTF2500
C   18  PENALIZED SHEKEL FUNCTION, M = 5  (N = 4)                          GLTF2510
C                                                                          GLTF2520
  180 CONTINUE                                                             GLTF2530
      M = 5                                                                GLTF2540
      GO TO 203                                                            GLTF2550
C                                                                          GLTF2560
C   19  PENALIZED SHEKEL FUNCTION, M = 7  (N = 4)                          GLTF2570
C                                                                          GLTF2580
  190 CONTINUE                                                             GLTF2590
      M = 7                                                                GLTF2600
      GO TO 203                                                            GLTF2610
C                                                                          GLTF2620
C   20  PENALIZED SHEKEL FUNCTION, M = 10  (N = 4)                         GLTF2630
C                                                                          GLTF2640
  200 CONTINUE                                                             GLTF2650
      M = 10                                                               GLTF2660
C                                                                          GLTF2670
  203 CONTINUE                                                             GLTF2680
      FUNZ = 0.D0                                                          GLTF2690
      DO 207 I = 1,M                                                       GLTF2700
         J = P20B(I)                                                       GLTF2710
         DO 205 J = 1,N                                                    GLTF2720
            S = S+(X(J)-P20A(I,J))**2                                      GLTF2730
  205    CONTINUE                                                          GLTF2740
         FUNZ = FUNZ - 1.D0/S                                              GLTF2750
  207 CONTINUE                                                             GLTF2760
      DO 209 I = 1,N                                                       GLTF2770
         IF (DABS(X(I)-5.D0).GT.5.D0)                                      GLTF2780
     1      FUNZ = FUNZ+PENFUN(X(I)-5.D0,5.D0,100.D0,2)                    GLTF2790
  209 CONTINUE                                                             GLTF2800
      RETURN                                                               GLTF2810
C                                                                          GLTF2820
C   21  PENALIZED THREE-DIMENSIONAL HARTMAN FUNCTION  (N = 3)              GLTF2830
C                                                                          GLTF2840
  210 CONTINUE                                                             GLTF2850
```

```
      S2 = S2+DI*DCOS((DI+1.D0)*X(2)+D1)                                GLTF1720
   95 CONTINUE                                                          GLTF1730
      FUNZ = FUNZ+S1*S2                                                 GLTF1740
      IF (DABS(X(1)).GT.10.D0) FUNZ = FUNZ+PENFUN(X(1),10.D0,100.D0,2)  GLTF1750
      IF (DABS(X(2)).GT.10.D0) FUNZ = FUNZ+PENFUN(X(2),10.D0,100.D0,2)  GLTF1760
      RETURN                                                            GLTF1770
C                                                                       GLTF1780
C   10  A FUNCTION WITH THREE ILL-CONDITIONED MINIMA, A=10  (N = 2)     GLTF1790
C                                                                       GLTF1800
  100 CONTINUE                                                          GLTF1810
      A = 1.D1                                                          GLTF1820
      B = 1.D-1                                                         GLTF1830
      GO TO 155                                                         GLTF1840
C                                                                       GLTF1850
C   11  A FUNCTION WITH THREE ILL-CONDITIONED MINIMA, A=10**2  (N = 2)  GLTF1860
C                                                                       GLTF1870
  110 CONTINUE                                                          GLTF1880
      A = 1.D2                                                          GLTF1890
      B = 1.D-2                                                         GLTF1900
      GO TO 155                                                         GLTF1910
C                                                                       GLTF1920
C   12  A FUNCTION WITH THREE ILL-CONDITIONED MINIMA, A=10**3  (N = 2)  GLTF1930
C                                                                       GLTF1940
  120 CONTINUE                                                          GLTF1950
      A = 1.D3                                                          GLTF1960
      B = 1.D-3                                                         GLTF1970
      GO TO 155                                                         GLTF1980
C                                                                       GLTF1990
C   13  A FUNCTION WITH THREE ILL-CONDITIONED MINIMA, A=10**4  (N = 2)  GLTF2000
C                                                                       GLTF2010
  130 CONTINUE                                                          GLTF2020
      A = 1.D4                                                          GLTF2030
      B = 1.D-4                                                         GLTF2040
      GO TO 155                                                         GLTF2050
C                                                                       GLTF2060
C   14  A FUNCTION WITH THREE ILL-CONDITIONED MINIMA, A=10**5  (N = 2)  GLTF2070
C                                                                       GLTF2080
  140 CONTINUE                                                          GLTF2090
      A = 1.D5                                                          GLTF2100
      B = 1.D-5                                                         GLTF2110
      GO TO 155                                                         GLTF2120
C                                                                       GLTF2130
C   15  A FUNCTION WITH THREE ILL-CONDITIONED MINIMA, A=10**6  (N = 2)  GLTF2140
C                                                                       GLTF2150
  150 CONTINUE                                                          GLTF2160
      A = 1.D6                                                          GLTF2170
      B = 1.D-6                                                         GLTF2180
C                                                                       GLTF2190
  155 CONTINUE                                                          GLTF2200
      XX = X(1)*X(1)                                                    GLTF2210
      YY = X(2)*X(2)                                                    GLTF2220
      R2 = XX+YY                                                        GLTF2230
      R4 = R2*R2                                                        GLTF2240
      R8 = R4*R4                                                        GLTF2250
      FUNZ = A*XX+YY-R4+B*R8                                            GLTF2260
      RETURN                                                            GLTF2270
C                                                                       GLTF2280
```

```
C                                                                       GLTF1150
   50 CONTINUE                                                          GLTF1160
      FUNZ = 0.D0                                                       GLTF1170
      DO 55 I = 1,5                                                     GLTF1180
         DI = DBLE(FLOAT(I))                                            GLTF1190
         FUNZ = FUNZ+DI*DCOS((DI+1.D0)*X(1)+DI)                         GLTF1200
   55 CONTINUE                                                          GLTF1210
      IF (DABS(X(1)).GT.10.D0) FUNZ = FUNZ+PENFUN(X(1),10.D0,100.D0,2)  GLTF1220
      RETURN                                                            GLTF1230
C                                                                       GLTF1240
C    4  A FOURTH ORDER POLYNOMIAL IN TWO VARIABLES  (N = 2)             GLTF1250
C                                                                       GLTF1260
   60 CONTINUE                                                          GLTF1270
      XX = X(1)*X(1)                                                    GLTF1280
      YY = X(2)*X(2)                                                    GLTF1290
      FUNZ = 0.25D0*XX*XX-0.5D0*XX+P4A*X(1)+0.500*YY                    GLTF1300
      RETURN                                                            GLTF1310
C                                                                       GLTF1320
C    5  A FUNCTION WITH A SINGLE ROW OF LOCAL MINIMA  (N = 2)           GLTF1330
C                                                                       GLTF1340
   70 CONTINUE                                                          GLTF1350
      FUNZ = 0.5D0*(P5A*X(1)*X(1)+1.D0-DCOS(2.D0*X(1)))+X(2)*X(2)       GLTF1360
      RETURN                                                            GLTF1370
C                                                                       GLTF1380
C    6  SIX-HUMP CAMEL FUNCTION  (N = 2)                                GLTF1390
C                                                                       GLTF1400
   80 CONTINUE                                                          GLTF1410
      XX = X(1)*X(1)                                                    GLTF1420
      YY = X(2)*X(2)                                                    GLTF1430
      FUNZ = ((XX/3.D0-(2.D0+P5A))*XX+4.D0)*XX+X(1)*X(2)                GLTF1440
     1     +4.D0*(YY-1.D0)*YY                                           GLTF1450
      RETURN                                                            GLTF1460
C                                                                       GLTF1470
C    7  TWO-DIMENSIONAL PENALIZED SHUBERT FUNCTION, BETA = 0  (N = 2)   GLTF1480
C                                                                       GLTF1490
   90 CONTINUE                                                          GLTF1500
      BETA = 0.D0                                                       GLTF1510
      GO TO 93                                                          GLTF1520
C                                                                       GLTF1530
C    8  TWO-DIMENSIONAL PENALIZED SHUBERT FUNCTION, BETA = 1/2  (N = 2) GLTF1540
C                                                                       GLTF1550
  100 CONTINUE                                                          GLTF1560
      BETA = 0.5D0                                                      GLTF1570
      GO TO 93                                                          GLTF1580
C                                                                       GLTF1590
C    9  TWO-DIMENSIONAL PENALIZED SHUBERT FUNCTION, BETA = 1  (N = 2)   GLTF1600
C                                                                       GLTF1610
  110 CONTINUE                                                          GLTF1620
      BETA = 1.D0                                                       GLTF1630
C                                                                       GLTF1640
   93 CONTINUE                                                          GLTF1650
      FUNZ = ((X(1)-P5A)**2+(X(2)-P5A)**2)*BETA                         GLTF1660
      SI = 0.D0                                                         GLTF1670
      CI = 0.D0                                                         GLTF1680
      DO 95 I = 1,5                                                     GLTF1690
         DI = DBLE(FLOAT(I))                                            GLTF1700
         SI = SI+DI*DCOS((DI+1.D0)*X(1)+DI)                             GLTF1710
```

```
      DATA PI /3.1415926535897932384600/                                GLTF0580
      DATA P1A /0.100/                                                  GLTF0590
      DATA P4A /0.100/                                                  GLTF0600
      DATA P5A /0.100/                                                  GLTF0610
      DATA P6A /0.100/                                                  GLTF0620
      DATA P9A,P9B /-1.4251284263197609708D0,-0.8003211004719731246600/ GLTF0630
      DATA P17A /1.275D0/                                               GLTF0640
      DATA P20A /4.00,1.00,8.00,6.00,3.00,2.00,5.00,8.00,6.00,7.00,     GLTF0650
     1           4.00,1.00,8.00,6.00,7.00,9.00,5.00,1.00,2.00,3.600,    GLTF0660
     2           4.00,1.00,8.00,6.00,3.00,2.00,3.00,8.00,6.00,7.00,     GLTF0670
     3           4.00,1.00,8.00,6.00,7.00,9.00,3.00,1.00,2.00,3.600/    GLTF0680
      DATA P20B /0.100,0.200,0.200,0.400,0.400,0.600,0.300,0.700,       GLTF0690
     1     0.500,0.500/                                                 GLTF0700
      DATA P21A /5.00,0.100,3.00,0.100,                                 GLTF0710
     1           10.00,10.00,10.00,10.00,                               GLTF0720
     2           30.00,35.00,30.00,35.00/                               GLTF0730
      DATA P21B /0.3689D0,0.4699D0,0.109100,0.03815D0,                  GLTF0740
     1           0.117000,0.4387D0,0.873200,0.5743D0,                   GLTF0750
     2           0.267300,0.747D0,0.554700,0.8828D0/                    GLTF0760
      DATA P21C /1.00,1.200,3.00,3.200/                                 GLTF0770
      DATA P21D /-69.00/                                                GLTF0780
      DATA P22A /10.00,0.05D0,3.00,17.00,3.00,10.00,3.50D0,8.00,        GLTF0790
     1           17.00,17.00,1.700,0.0500,3.500,0.100,10.00,10.00,      GLTF0800
     2           1.700,8.00,1.600,0.100,8.00,14.00,8.00,14.00/          GLTF0810
      DATA P22B /0.1312D0,0.2329D0,0.2348D0,0.4047D0,                   GLTF0820
     1           0.1696D0,0.4135D0,0.1451D0,0.8828D0,                   GLTF0830
     2           0.5569D0,0.8307D0,0.3522D0,0.8732D0,                   GLTF0840
     3           0.0124D0,0.3736D0,0.2883D0,0.5743D0,                   GLTF0850
     4           0.3736D0,0.1004D0,0.3047D0,0.1091D0,                   GLTF0860
     5           0.9991D0,0.9991D0,0.6650D0,0.0381D0/                   GLTF0870
      DATA P22C /1.00,1.200,3.00,3.200/                                 GLTF0880
      DATA P22D /-69.00/                                                GLTF0890
      DATA P36A,P36B,P36C,P36D /10.01,1.00,10.00,0.9800/                GLTF0900
      DATA P37A,P37B,P37C,P37D /10.00,1.00,10.00,0.9800/                GLTF0910
C                                                                       GLTF0920
C     PENALIZATION FUNCTION                                             GLTF0930
C                                                                       GLTF0940
      PENFUN (VAR,RAN,FACT,IEXP) = FACT*(DABS(VAR)-RAN)**IEXP           GLTF0950
C                                                                       GLTF0960
      GO TO (10,20,30,40,50,60,70,80,90,100,110,120,130,140,150,        GLTF0970
     1       160,170,180,190,200,210,220,230,240,250,260,270,280,       GLTF0980
     2       290,300,310,320,330,340,350,360,370), NPROB                GLTF0990
C                                                                       GLTF1000
C     1  A FOURTH ORDER POLYNOMIAL  (N = 1)                             GLTF1010
C                                                                       GLTF1020
   10 CONTINUE                                                          GLTF1030
      FUNCT = ((0.2500*X(1)*X(1)-0.500)*X(1)+P1A)*X(1)                  GLTF1040
      RETURN                                                            GLTF1050
C                                                                       GLTF1060
C     2  GOLDSTEIN SIXTH ORDER POLYNOMIAL  (N = 1)                      GLTF1070
C                                                                       GLTF1080
   20 CONTINUE                                                          GLTF1090
      AX = A(1)*X(1)                                                    GLTF1100
      FUNCT = ((AX-15.00)*AX+27.00)*AX*250.00                           GLTF1110
      RETURN                                                            GLTF1120
C                                                                       GLTF1130
C     3  ONE-DIMENSIONAL PENALIZED CHUBERT FUNCTION  (N = 1)            GLTF1140
```

```
      SUBROUTINE GLOMTF (NPROB, N, X, FUNZ)                              GLTF0010
C                                                                        GLTF0020
C     THE SUBROUTINE  GLOMTF  PROVIDES THE CODING OF 37 REAL-VALUED      GLTF0030
C     FUNCTIONS OF  N  REAL VARIABLES, TO BE USED, TOGETHER WITH THE     GLTF0040
C     SUBROUTINE  GLOMIP , TO DEFINE 37 TEST PROBLEMS FOR GLOBAL         GLTF0050
C     MINIMIZATION SOFTWARE.                                             GLTF0060
C                                                                        GLTF0070
C     THE SUBROUTINE  GLOMTF  RETURNS IN  FUNZ  THE FUNCTION VALUE       GLTF0080
C     AT THE POINT  X = (X1,X2,....,XN)  FOR THE FUNCTION DEFINED BY     GLTF0090
C     PROBLEM NUMBER  NPROB .                                            GLTF0100
C                                                                        GLTF0110
C     CALLING STATEMENT                                                  GLTF0120
C                                                                        GLTF0130
C        CALL GLOMTF (NPROB, N, X, FUNZ)                                 GLTF0140
C                                                                        GLTF0150
C     DESCRIPTION OF THE CALL PARAMETERS                                 GLTF0160
C        (THE FORTRAN IMPLICIT TYPE DEFINITION FOR INTEGERS IS USED.     GLTF0170
C        ALL NON-INTEGER PARAMETERS ARE DOUBLE-PRECISION).               GLTF0180
C                                                                        GLTF0190
C     NPROB   IS THE (INPUT) TEST-PROBLEM NUMBER                         GLTF0200
C                                                                        GLTF0210
C     N   IS THE (INPUT) DIMENSION OF THE PROBLEM                        GLTF0220
C                                                                        GLTF0230
C     X   IS AN (INPUT) N-VECTOR CONTAINING THE INDEPENDENT VARIABLES    GLTF0240
C                                                                        GLTF0250
C     FUNZ   IS THE (OUTPUT) VALUE AT  X  OF THE FUNCTION DEFINED BY     GLTF0260
C        PROBLEM NUMBER  NPROB .                                         GLTF0270
C                                                                        GLTF0280
C                                                                        GLTF0290
C                                                                        GLTF0300
      DOUBLE PRECISION X                                                 GLTF0310
      DOUBLE PRECISION FUNZ                                              GLTF0320
C                                                                        GLTF0330
      DOUBLE PRECISION PI                                                GLTF0340
      DOUBLE PRECISION P1A                                               GLTF0350
      DOUBLE PRECISION P4A                                               GLTF0360
      DOUBLE PRECISION P5A                                               GLTF0370
      DOUBLE PRECISION P6A                                               GLTF0380
      DOUBLE PRECISION P9A,P9B                                           GLTF0390
      DOUBLE PRECISION P17A                                              GLTF0400
      DOUBLE PRECISION P20A,P20B                                         GLTF0410
      DOUBLE PRECISION P21A,P21B,P21C,P21D                               GLTF0420
      DOUBLE PRECISION P22A,P22B,P22C,P22D                               GLTF0430
      DOUBLE PRECISION P36A,P36B,P36C,P36D                               GLTF0440
      DOUBLE PRECISION P37A,P37B,P37C,P37D                               GLTF0450
      DOUBLE PRECISION Y                                                 GLTF0460
      DOUBLE PRECISION AA,BI,YY,BETA,S1,S2,A0,H2,R4,R8,U,V,UU,VV,S       GLTF0470
      DOUBLE PRECISION RG,RP,H,FUND,RANGE                                GLTF0480
      DOUBLE PRECISION FACT,VAR,PENFUN,RAN                               GLTF0490
C                                                                        GLTF0500
      DIMENSION X(N)                                                     GLTF0510
C                                                                        GLTF0520
      DIMENSION P20A(10,4),P20B(10)                                      GLTF0530
      DIMENSION P21A(4,5),P21B(4,5),P21C(4)                              GLTF0540
      DIMENSION P22A(4,5),P22B(4,5),P22C(4)                              GLTF0550
      DIMENSION V(10)                                                    GLTF0560
C                                                                        GLTF0570
```

APPENDIX A4

A global optimization algorithm using stochastic differential equations

by F. Aluffi-Pentini, V. Parisi, F. Zirilli

(submitted to ACM Transactions on Mathematical Software).

A GLOBAL OPTIMIZATION ALGORITHM USING STOCHASTIC DIFFERENTIAL EQUATIONS

Filippo Aluffi-Pentini

Bari University

Valerio Parisi

2nd Rome University

Francesco Zirilli

Salerno University

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 n. DAJA-37-81-C-0740 with the University of Camerino, Italy.

Authors' Addresses:

F. Aluffi-Pentini, Dipartimento di Matematica, Università di Bari, 70125 Bari (Italy);
V. Parisi, Istituto di Fisica, 2[a] Università di Roma "Tor Vergata", Via Orazio Raimondo, 00173 (La Romanina) Roma (Italy);
F. Zirilli, Istituto di Matematica, Università di Salerno, 84100 Salerno (Italy).

ABSTRACT

SIGMA is a set of FORTRAN subprograms for solving the global optimization problem, which implement a method founded on the numerical solution of a Cauchy problem for stochastic differential equations inspired by quantum physics.

This paper gives a detailed description of the method as implemented in SIGMA, and reports on the numerical tests which have been performed while the SIGMA package is described in the accompanying Algorithm.

The main conclusions are that SIGMA performs very well on several hard test problems; unfortunately given the state of the mathematical software for global optimization it has not been possible to make conclusive comparisons with other packages.

Categories and Subject Descriptors:

G.1.0 [Numerical Analysis]: Optimization - ;
G.4 [Mathematical Software]: Algorithm analysis; certification and Testing.

General terms: Algorithms, Theory, Verification

Additional Key Words and Phrases: Global Optimization, Stochastic Differential Equations.

1. Introduction.

In [1] a method for solving the global optimization problem was proposed. The method associates a stochastic differential equation with the function whose global minimizer we are looking for.

The stochastic differential equation is a stochastic perturbation of a "steepest descent" ordinary differential equation and is inspired by quantum physics. In [1] the problem of the numerical integration of the stochastic equations introduced was considered and a suitable "stochastic" variation of the Euler method was suggested.

SIGMA is a set of FORTRAN subprograms implementing the above method.

In sect. 2 we describe the method as implemented in SIGMA; in sect. 3 we give a general description of the method and some details on the implementation; in sect. 4 some numerical experience on test problems is presented and in sect. 5 conclusions are given.

Unfortunately, given the state of the art of mathematical software in global optimization, it has not been possible to make conclusive comparisons with other packages.

The SIGMA package and its usage are described in the accompanying Algorithm.

2. The method.

Let $\mathbb{R}^N$ be the N-dimensional real euclidean space and let $f:\mathbb{R}^N \to \mathbb{R}$ be a real valued function, regular enough to justify the following considerations.

In this paper we consider the problem of finding a global minimizer of f, that is, the point $\underline{x}^* \in \mathbb{R}^N$ (or possible one of the points) such that

$$f(\underline{x}^*) \leq f(\underline{x}) \qquad \forall \underline{x} \in \mathbb{R}^N \tag{2.1}$$

and we propose a method introduced in [1] inspired by quantum physics to compute numerically the global minimizers of f by following the paths of a stochastic differential equation.

The interest of the global optimization problem both in mathematics and in many applications is well known and will not be discussed here.

We want just to remark here that the root-finding problem for the system $\underline{g}(\underline{x}) = \underline{0}$, where $\underline{g}:\mathbb{R}^N \to \mathbb{R}^N$ can be formulated as a global optimization problem considering the function $F(\underline{x}) = \|\underline{g}(\underline{x})\|_2^2$, where $\|\cdot\|_2$ is the euclidean norm in $\mathbb{R}^N$.*

Despite its importance and the efforts of many researchers the global optimization problem is still rather open and there is a need for methods with solid mathematical foundation and good numerical performance.

---

* The present authors have considered this idea both from the mathematical point of view (for a review see [2]) and from the point of view of producing good software (see [3], [4]). The method implemented in [3], [4] is inspired by classical mechanics, uses ordinary differential equations, and can be regarded as a method for global optimization.

Much more satisfactory is the situation for the problem of finding the local minimizers of f, where a large body of theoretical and numerical results exists; see for instance [5], [6] and the references given therein.

Ordinary differential equations have been used in the study of the local optimization problem or of the root finding problem by several authors; for a review see [2].

The above methods usually obtain the local minimizers or roots by following the trajectories of suitable ordinary differential equations. However, since the property (2.1) of being a global minimizer is a global one, that is, depends on the behaviour of f at each point of $\mathbb{R}^N$, and the methods that follow a trajectory of an ordinary differential equation are local, that is, they depend only on the behaviour of f along the trajectory, there is no hope of building a completely satisfactory method for global optimization based on ordinary differential equations.

The situation is different if we consider a suitable stochastic perturbation of an ordinary differential equation as explained in the following.

Let us first consider the (Ito) stochastic differential equation

$$d\underline{\xi} = -\nabla f(\underline{\xi})dt + \varepsilon d\underline{w} \tag{2.2}$$

where ∇f is the gradient of f and $\underline{w}(t)$ is a standard N-dimensional Wiener process, $\varepsilon \in \mathbb{R}$.

Equation (2.2) is known as the Smoluchowski-Kramers equation [7]; this equation is a singular limit of the Langevin's equation when the inertial terms are neglected.

The Smoluchowski-Kramers equation has been extensively used by solid state physicists and chemists to study physical phenomena such as atomic diffusion in crystals or chemical reactions.

In these applications (2.2) represents diffusion across potential barriers under the stochastic forces $\varepsilon d\underline{w}$, where $\varepsilon = \sqrt{\frac{2kT}{m}}$, T is the absolute temperature, k the Boltzmann constant, m a suitable mass coefficient, and f is the potential energy.

We assume that

$$\lim_{\|\underline{x}\|_2 \to \infty} f(\underline{x}) = +\infty \tag{2.3}$$

in such a way that:

$$\int_{\mathbb{R}^N} e^{-\alpha^2 f(\underline{x})}\, d\underline{x} < \infty \qquad \forall \alpha \in (\mathbb{R}\setminus\{0\}) \tag{2.4}$$

and that the minimizers of f are isolated and non degenerate.

It is well known that if $\underline{\xi}^\varepsilon(t)$ is the solution process of (2.2) starting from an initial point $\underline{x}_0$, the probability density function $p^\varepsilon(t,\underline{x})$ of $\underline{\xi}^\varepsilon(t)$ approaches as $t \to \infty$ the limit density $p_0^\varepsilon(\underline{x})$ where

$$p_0^\varepsilon(\underline{x}) = A_\varepsilon\, e^{-\frac{2}{\varepsilon^2} f(\underline{x})} \tag{2.5}$$

where A_ε is a normalization constant. The way in which $p^\varepsilon(t,\underline{x})$ for a class of one-dimensional systems approaches $p_0^\varepsilon(\underline{x})$ has been studied in detail by considering the spectrum of the corresponding Fokker-Planck operators in [8].

We note that p_ε is independent of $\underline{x}_0$ and that as $\varepsilon \to 0$ p_ε becomes more concentrated at the global minimizers of f. That is,

$$\lim_{t\to\infty} \underline{\xi}(t) = \underline{\xi}_\infty \quad \text{in law} \tag{2.6}$$

where $\underline{\xi}_\infty$ has a probability density given by (2.5) and

$$\lim_{\varepsilon\to 0} \underline{\xi}_\infty = \underline{\xi}_0 \quad \text{in law} \tag{2.7}$$

where $\underline{\xi}_0$ is a random variable having as its probability density a weighted sum of Dirac's deltas concentrated at the global minimizers of f. For example if $N = 1$ and f has two global minimizers x_1, x_2, with $\frac{d^2 f}{dx^2}(x_i) = c_i > 0$, $i = 1,2$, we have (in distribution sense)

$$\lim_{\varepsilon\to 0} p_\varepsilon(x) = \lambda\,\delta(x-x_1) + (1-\lambda)\,\delta(x-x_2) \tag{2.8}$$

where $\lambda = (1+\sqrt{c_2/c_1}\,)^{-1}$. In order to obtain the global minimizers of f as asymptotic values as $t \to \infty$ of a sample trajectory of a suitable system of stochastic differential equations it seems natural to try to perform the limit $t \to \infty$ (i.e. (2.6)) and the limit $\varepsilon \to 0$ (i.e. (2.7)) together.

That is, we want to consider:

$$d\underline{\xi} = -\nabla f(\underline{\xi})dt + \sigma(t)d\underline{w} \tag{2.9}$$

with initial condition

$$\underline{\xi}(0) = \underline{x}_0 \tag{2.10}$$

where

$$\lim_{t\to\infty} \sigma(t) = 0. \tag{2.11}$$

In physical terms condition (2.11) means that the temperature T is decreased to 0 (absolute zero) when $t \to \infty$, that is, the system is "frozen".

Since we want to end up in a global minimizer of f, that is, a global minimizer of the (potential) energy, the system has to be frozen very slowly (adiabatically). The way in which $\varepsilon(t)$ must go to zero, in order to have that when $t \to \infty$, the solution $\xi(t)$ of (2.9) becomes concentrated at the global minimizers of f, depends on f. In particular, it depends on the highest barrier in f to be overcome to reach the global minimizers.

This dependence has been studied using the adiabatic perturbation theory in [1]. Similar ideas in the context of combinatorial optimization have been introduced by Kirkpatrick, Gelatt, Vecchi in [9].

In this paper we restrict our attention to the numerical implementations of the previous ideas, that is, the computation of the global minimizers of f by following the paths defined by (2.9), (2.10), disregarding mathematical problems such as the difference between the convergence in law of $\xi(t)$ to a random variable concentrated at the global minimizers of f, and the convergence with probability one of the paths of $\xi(t)$ to the global minimizers of f.

We consider the problem of how to compute numerically these paths keeping in mind that we are not really interested in the paths, but only in their asymptotic values.

We discretize (2.9), (2.10) using the Euler method, that is $\xi(t_k)$ is approximated by the solution ξ_k of the following finite difference equations:

$$\xi_{k+1} - \xi_k = -h_k \nabla f(\xi_k) + \sigma(t_k)(w_{k+1} - w_k) \qquad k = 0,1,2, \ldots \tag{2.12}$$

$$\xi_0 = x_0 \tag{2.13}$$

where $t_0 = 0$, $t_k = \sum_{i=0}^{k-1} h_i$, $h_k > 0$, and $w_k = w(t_k)$, $k = 0,1,2, \ldots$.

The computationally cheap Euler step seems a good choice here since in order to obtain the global minimizers of f as asymptotic values of the paths $\xi(t)$ should go to zero very slowly when $t \to \infty$, and therefore a large number of time integration steps must be computed.

On the right hand side of (2.12) we add the random term $\sigma(t_k)(w_{k+1} - w_k)$ to the deterministic term $-h_k \nabla f(\xi_k)$, which is computationally more expensive (e.g. N+1 function evaluations if a forward-difference gradient is used), so that the effort spent in evaluating $\nabla f(\xi_k)$ is frequently lost.

In order to avoid this inconvenience we substitute the gradient $\nabla f(\xi)$ with a "random gradient" as follows. Let r be an N-dimensional random vector of length 1 uniformly distributed on the N-dimensional unit sphere. Then for any given (non-random) vector $v \in \mathbb{R}^N$ its projection along r is such that:

$$N \cdot E(\langle r, v \rangle r) = v \tag{2.14}$$

where $E(\cdot)$ is the expected value, and $\langle \cdot, \cdot \rangle$ is the euclidean inner product in $\mathbb{R}^N$.

So that in order to save numerical work (i.e. functions evaluations) in (2.12) we substitute $\nabla f(\xi_k)$ with the "random gradient"

(2.15) $$\underline{\ell}(\underline{\xi}_k) = N < \underline{r}, \nabla f(\underline{\xi}_k) > \underline{r}.$$

We note that since $\frac{1}{N}\underline{\ell}(\underline{\xi}_k)$ is the directional derivative in the direction $\underline{r}$, it is computationally much cheaper (e.g. when forward differences are used, only 2 function evaluations are needed to approximate $\underline{\ell}(\underline{\xi})$). Therefore, the paths are computed approximating $\underline{\xi}(t_k)$ with the solution $\underline{\xi}_k$ of the following differences equations:

(2.16) $$\underline{\xi}_{k+1} - \underline{\xi}_k = -h_k \underline{\ell}(\underline{\xi}_k) + \underline{\sigma}(t_k)(\underline{w}_{k+1} - \underline{w}_k) \quad k = 0,1,2, \ldots$$

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

where $\underline{\ell}(\underline{\xi}_k)$ is a finite difference (forward or central) approximation to $\underline{\ell}(\underline{\xi}_k)$.

The complete algorithm is described in the next section.

3. The complete algorithm.

We give in sect. 3.1 a general description of the algorithm, while implementation details are given in sect. 3.2.

3.1. General description of the algorithm.

The basic time-integration step (eq. (2.16) and sect. 3.2.1) is used to generate a fixed number N_{TRAJ} of trajectories, which start at time zero from the same initial conditions with possibly different values of $x(0)$ (note that even if the starting values $x(0)$ are equal the trajectories evolve differently due to the stochastic nature of the integration steps).

The trajectories evolve (simultaneously but independently) during an "observation period" having a given duration (sect. 3.2.5), and within which the noise coefficient of each trajectory is kept at a constant value β, while the values of the steplength h_k and of the spatial discretization increment Δx_k for computing the random gradient (eq. (2.17) and sect. 3.2.2) are automatically adjusted for each trajectory by the algorithm (sects. 3.2.3 and 3.2.4).

At the end of every observation period the corresponding trajectories are compared, one of them is discarded (and will not be considered any more), all other trajectories are naturally continued in the next observation period, and one of the trajectories is selected for "branching" (sect. 3.2.6), that is for generating also a second continuation trajectory differing from the first one only in the starting values for β and x_k (sect. 3.2.7), and which is considered as having the same "past history" of the first one.

The set of simultaneous trajectories is considered as a single trial, which is stopped as described in sect. 3.2.8, and is repeated a number of times with different operating conditions (sect. 3.2.9).

The stopping criteria for the complete algorithm are described in sect. 3.2.10.

The use of an admissible region for the x-values is described in sect. 3.2.11, scaling is described in sect. 3.2.12, and criteria for numerical equality in sect. 3.2.13.

3.2. Implementation details.

3.2.1 The time-integration step.

The basic time-integration step (eq. (2.16)) is used, for the trajectory under consideration, in the form

$$\underline{\xi}_{k+1} = \underline{\xi}_k - h_k \underline{\zeta}(\underline{\xi}_k) + \varepsilon_p \sqrt{h_k}\, \underline{u}_k \quad (k = 0,1,2, \ldots) \tag{3.2.1.1}$$

where h_k and ε_p are the current values of the steplength and of the noise coefficient (the noise coefficient has a constant value ε_p throughout the current observation period (sect. 3.1)); $\underline{u}_k$ is a random vector sample from an N-dimensional standard Gaussian distribution, and

$$\sqrt{h_k}\, \underline{u}_k = \underline{w}_{k+1} - \underline{w}_k$$

due to the properties of the Wiener process.

The computation of the finite-differences random gradient $\underline{\zeta}(\underline{\xi}_k)$ is described in the next section.

The basic step (3.2.1.1) is actually performed in two half-steps

$$\underline{\xi}'_k = \underline{\xi}_k - h_k \underline{\nu}(\underline{\xi}_k) \qquad \text{(first half-step)} \tag{3.2.1.2}$$

and

$$\underline{\xi}_{k+1} = \underline{\xi}'_k + \sigma_p \sqrt{h_k}\, \underline{u}_k \qquad \text{(second half-step)} \tag{3.2.1.3}$$

Both half-steps depend on h_k while the first depends also on the current value Δx_k of the spatial discretization increment used in computing $\underline{\nu}(\underline{\xi}_k)$.

Either half-step can be rejected if deemed not satisfactory, as described in sect. 3.2.3.

3.2.2 The finite-differences random gradient.

Given the current value Δx_k of the spatial discretization increment for the trajectory under consideration, we consider the random increment vector

$$\underline{s}_k = \Delta x_k \cdot \underline{r}_k$$

where $\underline{r}_k$ is a random sample of a vector uniformly distributed on the unit sphere in $\mathbb{R}^N$, the forward and central differences

$$\begin{cases} \Delta^F f_k = f(\underline{\xi}_k + \underline{s}_k) - f(\underline{\xi}_k) \\ \Delta^C f_k = \frac{1}{2}[f(\underline{\xi}_k + \underline{s}_k) - f(\underline{\xi}_k - \underline{s}_k)] \end{cases} \tag{3.2.2.1}$$

the forward- and central-differences directional derivatives

$$\delta^F_k = \Delta^F f_k / \Delta x_k \qquad \delta^C_k = \Delta^C f_k / \Delta x_k \tag{3.2.2.2}$$

4. Numerical Testing.

SIGMA has been numerically tested on a number of test problems run on two computers. The test problems are described in sect. 4.1, the computers in sect. 4.2 and some numerical results are reported in sect. 4..

4.1. Test problems.

The set of test problems is fully described in [10] together with the initial points; the test problems are:

1. A fourth-order polynomial (N = 1)
2. Goldstein sixth-order polynomial (N = 1)
3. One-dimensional penalized Shubert function (N = 1)
4. A fourth-order polynomial in two variables (N = 2)
5. A function with a single row of local minima (N = 2)
6. Six-hump camel function (N = 2)
7. Two-dimensional penalized Shubert function $\beta = 0$ (N = 2)
8. Two-dimensional penalized Shubert function $\beta = 0.5$ (N = 2)
9. Two-dimensional penalized Shubert function $\beta = 1$ (N = 2)
10. A function with three ill-conditioned minima a = 10 (N = 2)
11. A function with three ill-conditioned minima a = 100 (N = 2)
12. A function with three ill-conditioned minima a = 1000 (N = 2)
13. A function with three ill-conditioned minima a = 10000 (N = 2)
14. A function with three ill-conditioned minima $a = 10^5$ (N = 2)
15. A function with three ill-conditioned minima $a = 10^6$ (N = 2)
16. Goldstein-Price function (N = 2)
17. Penalized Branin function (N = 2)
18. Penalized Shekel function M = 5 (N = 4)

a) Relative difference criterion

$$|x-y| \leq \tau_{REL} \quad (|x| + |y|)/2$$

b) Absolute difference criterion

$$|x-y| \leq \tau_{ABS}$$

where τ_{REL} and τ_{ABS} are given non-negative tolerances.

Let λ_1 be the largest eigenvalue of the (symmetric and non-negative definite) matrix C.

We adopt the updating matrix

$$F_A = \beta\lambda_1 I - C$$

where I is the N×N identity matrix, $\beta > 1$ ($\beta = 1.3$ in the present implementation), and we obtain the updated value A′ of A by means of the formula

$$A' = \alpha A F_A$$

where α is a normalization factor such that the sum of the squares of the elements of A′ is equal to N (as in the identity matrix).

The matrix F_A seems one of the possible reasonable choices, since it is positive definite for $\beta > 1$, it has the same set of eigenvectors as C, its eigenvalue spectrum is obtained from the spectrum of C by reflection around $\lambda = \frac{\beta\lambda_1}{2}$, and it therefore acts in the right direction to counter the ill-conditioning of $\tilde{f}$.

The magnitude of the counter-effect depends on β: the adopted value has been experimentally adjusted.

The updated bias vector $\underline{b}'$ is chosen in order that the scaling at $\underline{x}$ does not alter $\tilde{\underline{x}}$, i.e. in order that

$$A'\underline{x} + \underline{b}' = A\underline{x} + \underline{b}.$$

3.2.13 Criteria for numerical equality.

The following two criteria are used in a number of places in the algorithm to decide if two given numbers x and y are sufficiently close to each other (within given tolerances) to be considered "numerically equal".

We consider (for each trajectory) the rescaled variable $\underline{\tilde{x}} = A\underline{x} + \underline{b}$, where A is the rescaling matrix and $\underline{b}$ is a bias vector, and, instead of $f(\underline{x})$, we minimize with respect to $\underline{\tilde{x}}$ the function $\tilde{f}(\underline{\tilde{x}}) = f(\underline{x}) = f(A\underline{x} + \underline{b})$, and we try to counter the ill-conditioning of f with respect to $\underline{x}$ by suitably adjusting A (and $\underline{b}$ is adjusted in order not to alter $\underline{x}$).

The updating of A is obtained by means of an updating matrix F_A, and if performed at the end of an observation period if sufficient data are available (see below), and if the number of elapsed observation periods is not less than a given number K_{pasca}, and greater than $7N$).

The updating matrix F_A is computed as described below, keeping in mind that the random gradients are the only simply-usable data on the behavior of f computed by the algorithm.

Let $\underline{\tilde{g}}_{(i)}$, $i = 1,2, \ldots, N_g$, be the column vectors of the components of all the N_g finite-difference random gradients $\underline{\tilde{g}}$ ($\underline{\tilde{g}}^F$ or $\underline{\tilde{g}}^C$) evaluated along the trajectory (also for rejected steps) from the last scaling.

If sufficient data are available (i.e. if $N_g \geq 2N^2$) we compute the average

$$\underline{\bar{\tilde{g}}} = \frac{1}{N_g} \sum_{i=1}^{N_g} \underline{\tilde{g}}_{(i)}$$

and the estimated covariance matrix

$$C = \frac{1}{N_g} \sum_{i=1}^{N_g} [(\underline{\tilde{g}}_{(i)} - \underline{\bar{\tilde{g}}})(\underline{\tilde{g}}_{(i)} - \underline{\bar{\tilde{g}}})^T]$$

which seems to be a reasonable indicator, given the available data, of the average ill-conditioning of f, as having the larger eigenvalues associated with the directions along which the second directional derivatives of f are, on the average, larger.

We note that each integration step can be rejected only a finite number of times, each observation period lasts a finite number of accepted integration steps, and there is a finite number of observation periods in a trial; since a finite number of trials is allowed, the algorithm will stop after a finite total number of steps and of function evaluations.

3.2.11 Admissible region for the x-values.

In order to help the user in trying to prevent computation failures (e.g. overflow) the present implementation of the method gives the possibility of defining (for any given problem and machine dynamic range, and based on possible analytical or experimental evidence) an admissible region for the x-values (hopefully containing the looked-for global minimizer) within which the function values may be safely computed. We use an N-dimensional interval

$$R_i^{MIN} \leq x_i \leq R_i^{MAX}, \quad i = 1,2, \ldots, N,$$

where the interval boundaries must be given before trial start.

Outside the admissible region the function $f(x)$ is replaced by an exponentially increasing function, in such a way that the values of f and of the external function are matched at the boundary of the region.

3.2.12 Scaling.

In order to make ill-conditioned problems more tractable, rescaling is performed by the algorithm as follows.

the preceding trial, according to the outcome (stopping condition) of the preceding trial and to the number t of trials performed from algorithm start, as compared to the given maximum number of trials N_{TRIAL}

successful stop: $\alpha = 10^{3}$

unsuccessful uniform stop:

$\alpha = 10$ if $t = [[(2/5)\ N_{TRIAL}]]$

$\alpha = 10^{-1}$ otherwise,

where $[[x]]$ is the smallest integer not smaller than x

unsuccessful non-uniform stop: $\alpha = 10^{-4}$

The initial point $\underline{x}_0$ is selected as follows:

if $t = [[(2/5)\ N_{TRIAL}]]$ take the value of $\underline{x}_0$ at algorithm start

otherwise take $\underline{x}_0 = \underline{x}_{OPT}$

where $\underline{x}_{OPT}$ is the current best minimizer found so far from algorithm start.

All other initial values are those of the first trial, except the initial values of h and λ_k which are the values reached at the end of the preceding trial.

3.2.10 Stopping criteria for the algorithm.

The complete algorithm is stopped, at the end of a trial, if a given number N_{SUC} has been reached of uniform trial stops all at the current f_{OPT} level, or in any case if a maximum given number N_{TRIAL} of trials has been reached.

Success is claimed by the algorithm if at least one uniform stop occurred at the current f_{OPT} level.

and the best minimum function falue f_{OPT} found so far from algorithm start: if f_{TEMIN} and f_{OPT} satisfy at least one of the above criteria, with the same tolerances, the trial is considered successful at the level f_{OPT}; otherwise the trial is again considered unsuccessful.

Checking of the stopping criteria is activated only if a minimum given number N_{PMIN} of observation periods has been reached.

3.2.9 Characteristics of the successive trials.

The operating conditions which are changed when another trial is started are:

- seed of the random number generator
- maximum duration of the trial
- policy for choosing c_p for the second continuation of a branched trajectory
- value of c_p at trial start
- initial point $\underline{x}_0$.

The maximum duration of a trial, i.e. the maximum number N_{PMAX} of observation periods, is obtained as follows:

if the preceding trial had a uniform ston (sect. 3.2.8) take the value of the preceding trial

otherwise take a value obtained by adding to the preceding value a fixed given increment I_{NPMAX}.

The policy for selecting c_p for the second continuation of a branched trajectory was described in sect. 3.2.7.

The value of c_p at the start of a new trial is obtained by means of a multiplicative updating factor α applied to the starting value of

The updating factor $F_{\cdot}$ for $\cdot_p$ is as follows:

for the first trial and for any trial following an unsuccessful trial

$F = 10^{x-\frac{1}{2}}$ where x is a random sample from a standard normal distribution

for all other trials

$F = 2^{y-\frac{1}{2}}$ where y is a random sample from a standard Cauchy distribution, i.e. with density

$f(y) = 1/(\pi(1+y^2))$

The updating factor for $\cdot x_k$ is:

$F_{\cdot x} = 10^{3z}$ where z is a random sample from a standard normal distribution.

3.2.8 Stopping criteria for a trial.

A trial is stopped, at the end of an observation period, and after having discarded the worst trajectory, if all the final function values of the remaining trajectories (possibly at different points x) are "numerically equal", i.e. if the maximum, f_{TPMAX}, and the minimum, f_{TPMIN}, among the trial final values satisfy at least one of the criteria in sect. 3.2.13, the relative difference criterion with a given stopping tolerance $\cdot_{\text{REL}}$ and/or the absolute difference criterion with given stopping tolerance $\cdot_{\text{ABS}}$ ("uniform stop at the level f_{TPMIN}").

The trial is also anyway stopped, at the end of the observation period, if a maximum given number N_{PMAX} of observation periods has been reached.

In the latter case the trial is considered unsuccessful, while in the former case a comparison is made between the final value f_{TPMIN}

From the point of view of the noise coefficient c_p a trajectory with larger c_p is considered better if the comparison is made in an early observation period (as long as $k_p < M_p \cdot I_b$, where k_p is the number of elapsed observation periods, and M_p, I_b are defined below) and worse otherwise.

A basic partial ordering of the trajectories is first obtained on the basis of past function values, and a final total ordering is then obtained, if needed, by suitably exploiting the noise-based ordering.

The discarded trajectory is always the worst in the ordering, while the trajectory selected for branching is usually not the best one, to avoid to be stuck in a non-global minimum.

Normal branching is performed on the trajectory which, in the ordering, occupies the place I_b (a given integer); exceptional branching, where the best trajectory is selected, occurs for the first time at the end of observation period k_{po}, and then every M_p periods (k_{po} and M_p are given integers); i.e. exceptional observation periods are those numbered

$$k_p = k_{po} + M_p \cdot j \quad (j = 0, 1, 2, \ldots)$$

3.2.7 The second continuation of a branched trajectory.

While the first (unperturbed) continuation of a trajectory that undergoes branching starts with the current values of c_p and Δx_k, the second continuation starts with values obtained by means of multiplicative random updating factors applied to the current values.

In phase 6a: $\quad = 0.1$

We finally remark that h_k and x_k are bounded by suitable constants to avoid computational failures.

5.2.5 Duration of the observation period.

The duration of observation period numbered k_p from trial start, defined as the number N_{hp} of time integration steps in period k_p, is computed as a function of k_p by means of a formula which must be chosen before algorithm start among the following three formulas:

1) $N_{hp} = 1 + [\log_2(k_p)]$ ("short" duration)

2) $N_{hp} = [k_p^{\frac{1}{2}}]$ ("medium-size" duration)

3) $N_{hp} = k_p$ ("long" duration)

where $k_p = 1, 2, \ldots$, and $[x]$ is the largest integer not greater than x.

5.2.6 Trajectory selection.

In order to decide, at the end of an observation period, which trajectory is to be discarded, and which one should be selected for branching, we compare the trajectories on the basis of the values of their noise coefficient in the observation period, and of the function values obtained from trial start.

From the point of view of past function values a trajectory is considered better than another if it has attained a lower function value than the other (excluding a possible initial part common to both trajectories).

We test f_k and $\hat{f}_k = f_k + \Delta f_k$ for numerical equality according to the relative difference criterion (sect. 3.2.13) with tolerances $\tau_{R1} = 10^{-11}$ and $\tau_{R2} = 10^{-5}$, and take

$\sigma = 2$ if f_k and $\hat{f}_k$ are "equal" within τ_{R1}

$\sigma = \frac{1}{2}$ if f_k and $\hat{f}_k$ are not "equal" within τ_{R2}

$\sigma = 1$ otherwise.

The interval $(10^{-11}, 10^{-5})$ has been adopted since it contains both the square root and the cubic root of the machine precision of most computers in double precision (the square root is appropriate for forward differences, while the cubic root is appropriate for central differences).

Updating factors γ for h_k

In phase 4a:

$\gamma = 1/1.05$ for the first attempt to the first half-step

$\gamma = \frac{1}{2}$ for the second attempt

$\gamma = 1/10$ for all other attempts

In phase 5 the value of γ depends on the current number a of accepted time integration steps in the current observation period, and on the current total number r of half-steps rejected so far in the current trial (excluding those possible rejected while attempting the first step).

If $r > 0$

$\gamma = 1$ (if $a \leq 2r$)

$\gamma = 1.1$ (if $2r < a \leq 3r$)

$\gamma = 2$ (if $3r < a$)

If $r = 0$

$\gamma = 2$ (if $a = 1$)

$\gamma = 10$ (if $a > 1$)

9a. If the half-step is rejected: reject also the first half-step, update (decrease) h_κ, and go back to 1.

9b. Otherwise: accept the whole step and try the next one.

Note however that if the same half-step is rejected too many times the half-step is nevertheless accepted in order not to stop the algorithm; this is not too harmful since several trajectories are being computed, and a "bad" one will be eventually discarded (in the present implementation the bound is given explicitly for the first half-step (50 times), and implicitly for the second half-step (if h_κ becomes smaller than 10^{-30})).

3.2.4 The updating of h_κ and Δx_κ.

The time-integration steplength h_κ and the spatial discretization increment Δx_κ for the trajectory under consideration are updated while performing the integration step, as described in the preceding section.

Updating is always performed by means of a multiplicative updating factor which is applied to the old value to obtain the new one.

The magnitude of the updating factors, as used in the various phases of the sequence in the preceding sect. 3.2.3, is as follows:

Updating factors ρ for Δx_κ

In phase 1b: ρ = 10

In phase 2a: ρ = 10

In phase 4b: ρ = 10^{-1}

In phase 3 the value of ρ depends on the magnitude of the current estimated function increment $\Delta f_\kappa = \rho_\kappa \Delta x_\kappa$ (where ρ_κ is ρ_k^f or ρ_k^c as appropriate), and the function value $f_\kappa = f(x_\kappa)$.

All attempts are with the current (i.e. updated) values of h_k and Δx_k.

The sequence of attempts is as follows:

1. Pick up a random unit vector $\underline{r}_k$.

1a. Compute the random increment $\underline{s}_k$ (sect. 3.2.2).

1b. If $\underline{s}_k$ (and therefore Δx_k) is too small (i.e. if the square of the euclidean norm of the difference between the computed values of $\underline{f}_k + \underline{s}_k$ and $\underline{f}_k$ is zero, due to the finite arithmetics of the machine): update (increase) Δx_k and go back to 1a.

2. Compute $\underline{g}_k^F$ (eq. (3.2.2.2)).

2a. If the computed value of $(\underline{g}_k^F)^2$ is zero (due to the finite arithmetics): update (increase) Δx_k and go back to 1a.

3. Compute the first half-step with $\underline{g}_k^F$.

Compute $\Delta' f_k$ (eq. (3.2.3.1)).

3a. If $\Delta' f_k \leq |p_k^F| \Delta x_k$

accept the first half-step and jump to 5.

4. Compute the first half-step with $\underline{g}_k^C$ to check the appropriateness of Δx_k.

Compute $\Delta' f_k$ (eq. (3.2.3.1)).

4a. If $\Delta' f_k > p_k^F - p_k^C | \Delta x_k$

reject the half-step, update (decrease) h_k, and go back to 1.

4b. Otherwise: accept the half-step, and update (decrease) Δx_k.

5. Update (increase) h_k.

Update (decrease) Δx_k.

6. Compute the second half-step.

Compute $\Delta'' f_k$ (eq. (3.2.3.2)).

and the forward- and central-differences random gradients

$$(3.2.2.3) \qquad \underset{\sim k}{\overset{F}{\Delta}} = N \, \rho_k^F \, \underset{\sim k}{r} \qquad \underset{\sim k}{\overset{C}{\Delta}} = N \, \rho_k^C \, \underset{\sim k}{r}$$

We use $\underset{\sim k}{\overset{F}{\Delta}}$ or $\underset{\sim k}{\overset{C}{\Delta}}$ for $\underset{\sim}{\Delta}(\underset{\sim k}{\xi})$ in the first half-step as described in the next section.

3.2.3 Accepting and rejecting the half-steps.

The computation of the first half-step can be attempted with the forward- or central-differences random gradient ($\underset{\sim k}{\overset{F}{\Delta}}$ or $\underset{\sim k}{\overset{C}{\Delta}}$ eq. (3.2.2.3)) as described below.

In either case the half-step is accepted or rejected according to the function increment

$$(3.2.3.1) \qquad \Delta' f_k = f(\underset{\sim k}{\xi'}) - f(\underset{\sim k}{\xi})$$

Since $\Delta' f_k$ should be non-positive for a sufficiently small value of h_k the half-step is rejected if $\Delta' f_k$ is "numerically positive", i.e. larger than a given positive small tolerance.

The second half-step is rejected if the corresponding function increment

$$(3.2.3.2) \qquad \Delta'' f_k = f(\underset{\sim k+1}{\xi}) - f(\underset{\sim k}{\xi'})$$

is positive and too large (greater than 100 ε_p in the present implementation).

The sequence of attempts affects the updating of h_k and Δx_k as described below; the amount of the updating is described in sect. 3.2.4.

19. Penalized Shekel function M = 7 (N = 4)

20. Penalized Shekel function M = 10 (N = 4)

21. Penalized three-dimensional Hartman function (N = 3)

22. Penalized six-dimensional Hartman function (N = 6)

23. Penalized Levy-Montalvo function, type 1 (N = 2)

24. Penalized Levy-Montalvo function, type 1 (N = 3)

25. Penalized Levy-Montalvo function, type 1 (N = 4)

26. Penalized Levy-Montalvo function, type 2 (N = 5)

27. Penalized Levy-Montalvo function, type 2 (N = 8)

28. Penalized Levy-Montalvo function, type 2, (N = 10)

29. Penalized Levy-Montalvo function, type 3, range 10 (N = 2)

30. Penalized Levy-Montalvo function, type 3, range 10 (N = 3)

31. Penalized Levy-Montalvo function, type 3, range 10 (N = 4)

32. Penalized Levy-Montalvo function, type 3, range 5 (N = 5)

33. Penalized Levy-Montalvo function, type 3, range 5 (N = 6)

34. Penalized Levy-Montalvo function, type 3, range 5 (N = 7)

35. A function with a cusp-shaped minima (N = 5)

36. A function with a global minimum having a small region of attraction a = 100 (N = 2)

37. A function with a global minimum having a small region of attraction a = 10 (N = 5)

We used the above functions, and the standard initial points as they are coded in the subroutines GLOMTF and GLOMIP, which are available in [10].

4.2. Test computers.

We considered two typical machines of "large" and "small" dynamic range, that is, with 11 and 8 bits for the exponent (biased or signed) of double precision numbers, and corresponding dynamic range of about $10^{\pm 308}$ and $10^{\pm 38}$. The tests were actually performed on:

— UNIVAC 1100/82 with EXEC8 operating system and FORTRAN (ASCII) computer (level 10R1) ("large" dynamic range)

— D.E.C. VAX 11/750 with VMS operating system (vers. 3.0) and FORTRAN compiler (vers. 3) ("small" dynamic range)

4.3. Numerical results.

Numerical results of running SIGMA on the above problems and on the above machines are described below. All results were obtained under the following operating conditions.

The easy-to-use driver subroutine SIGMA1 (described in the accompanying algorithm) was used, with $N_{SUC} = 1,2,3,4,5$. All numerical values used for the parameters are set in the driver SIGMA1 and in the other subroutines which are described in the accompanying Algorithm.

All numerical results are reported on Tables 1, 2, and 3. Table 1 reports some performance data (i.e. output indicator IOUT and number of functions evaluations) as obtained from SIGMA output for each of the 37 test problems and for the testing both on the "large" and "small" dynamic range machines. In order to evaluate the performance of SIGMA we consider all the cases in which the program claimed a success (output indicator $IOUT > 0$) or a failure ($IOUT \leq 0$) and — by comparing the final point

with the known solutions — we identify the cases in which such a claim is clearly incorrect (i.e. success claim when the final point is not even approximately close to the known solution, or failure claim when the final point is practically coincident with the known solution). It is also meaningful to consider all the cases in which a computational failure due to overflow actually occurrs at any point of the iteration.

Table 2 and Table 3 report for each problem and summarized for all problems data concerning the effectiveness, dependability and robustness — in the form of total numbers of correctly claimed successes, correctly claimed failures, incorrect success or failure claims and total number of overflows — for the two machines.

TABLE 1

UNIVAC

N_{SUC} =		1		2		3		4		5	
NPROB	N	Nf	Ie	Nf	Ie	Nf	Ie	Nf	Ie	Nf	Ie
1	1	5,588	0	11,467	0	23,067	0	32,520	0	58,751	0
2	1	3,254	0	9,509	0	20,893	0	32,910	0	72,015	0
3	1	8,638	0	17,741	0	23,814	0	57,187	0	67,621	0
4	2	6,594	0	15,898	0	30,589	0	69,489	0	101,633	0
5	2	12,680	0	23,221	0	38,362	0	95,423	0	104,391	0
6	2	2,697	0	8,343	0	19,660	0	57,728	0	78,090	0
7	2	32,185	0	35,256	0	49,153	0	59,983	0	139,675	0
8	2	5,600	0	347,039	0	348,301	0	359,642	0	392,466	0
9	2	6,180	0	83,625	0	470,130	0	699,767	0	701,051	0
10	2	3,596	0	6,731	0	12,958	0	61,753	0	66,855	0
11	2	3,191	0	8,384	0	23,196	0	40,808	0	56,958	0
12	2	4,799	0	7,296	0	18,902	0	29,315	0	47,216	0
13	2	7,105	0	10,287	0	20,605	0	27,838	0	43,505	0
14	2	6,671	0	10,654	0	15,102	0	31,322	0	47,051	0
15	2	7,747	0	11,631	0	16,227	0	23,587	0	38,362	0
16	2	16,021	0	26,560	0	58,401	0	67,865	0	115,350	0
17	2	2,700	0	6,670	0	14,388	0	28,275	0	80,826	0
18	4	4,674	0	16,556	0	101,828	0	209,177	0	282,950	0
19	4	4,759	0	54,559	0	131,350	0	224,028	0	306,327	0
20	4	9,955	0	,092	0	26 ,616	0	78, 3	0	327,392	0
21	3	3,416	0	,520	0	2 ,.				88, 5	0
22		4,729	0	3 8	0	2 ,31				52,364	0
23		11,88	0	( )	0					92, 1	0

Table 1 (continued)

UNIVAC (continued)

N_{SUC} =		1		2		3		4		5	
NPROB	N	Nf	Ie	Nf	Ie	Nf	Ie	Nf	Ie	Nf	Ie
24	3	8,099	0	36,057	0	47,619	0	69,901	0	92,104	0
25	4	11,954	0	54,212	0	71,655	0	97,724	0	191,722	0
26	5	43,083	0	284,104	0	347,056	0	450,102	0	464,611	0
27	8	2,324	0	21,124	0	75,728	0	635,990	0	654,436	0
28	10	50,975	0	426,171	0	454,808	0	474,323	0	479,817	0
29	2	25,462	0	35,675	0	98,944	0	111,447	0	167,728	0
30	3	15,734	0	113,789	0	177,970	0	257,904	0	286,273	0
31	4	11,516	0	143,757	0	208,217	0	264,834	0	296,663	0
32	5	50,911	0	176,840	0	275,852	0	357,089	0	679,442	0
33	6	53,178	0	102,652	0	272,642	0	303,267	0	454,543	0
34	7	14,594	0	298,256	0	357,878	0	409,949	0	520,641	0
35	5	33,635	0	50,348	0	70,105	0	127,091	0	183,864	0
36	2	3,102	0	10,176	0	23,283	0	72,931	0	79,481	0
37	5	6,938	0	12,469	0	25,175	0	64,639	0	92,407	0

Table 1 (continued)

N_{SUC} =		VAX 1		2		3		4		5	
NPROB	N	Nf	Ie	Nf	Ie	Nf	Ie	Nf	Ie	Nf	Ie
1	1	7,522	0	12,657	0	21,643	0	31,787	0	46,954	0
2	1	3,131	0	6,542	0	14,171	0	27,023	0	70,953	0
3	1	11,526	0	15,342	0	20,457	0	57,342	0	67,423	0
4	2	9,265	0	17,713	0	28,667	0	80,161	0	144,719	0
5	2	12,094	0	19,716	0	36,426	0	59,214	0	100,336	0
6	2	4,650	0	11,040	0	25,772	0	57,087	0	62,099	0
7	2	10,543	0	44,408	0	82,833	0	130,859	0	156,208	0
8	2	27,044	0	76,348	0	189,195	0	521,474	0	604,401	0
9	2	24,348	0	35,885	0	71,593	0	165,393	0	225,842	0
10	2	4,114	0	9,959	0	19,363	0	42,409	0	91,572	0
11	2	3,254	0	7,901	0	12,795	0	28,450	0	42,615	0
12	2	6,711	0	9,949	0	18,191	0	28,788	0	44,896	0
13	2	6,771	0	11,031	0	15,508	0	22,629	0	39,765	0
14	2	6,208	0	9,443	0	17,719	0	23,579	0	39,721	0
15	2	6,313	0	13,581	0	17,631	0	30,648	0	42,360	0
16	2	5,439	0	10,491	0	24,055	0	80,137	0	101,441	0
17	2	2,790	0	11,006	0	18,444	0	51,305	0	61,100	0
18	4	2,446	0	36,252	0	129,264	0	269,925	0	290,805	0
19	4	4,778	0	19,951	0	44,198	0	109,747	0	273,679	0
20	4	4,741	0	11,312	0	24,947	0	78,820	0	125,407	0
21	3	4,334	0	27,816	0	44,893	0	82,640	0	150,742	0
22	6	3,975	0	[illegible]	0	[illegible]		[illegible]		[illegible]	0
23		[illegible]	0	[illegible]		[illegible]		[illegible]		[illegible]	

Table 1 (continued)

VAX (continued)

N_{SUC} =		1		2		3		4		5	
NPROB	N	Nf	Ie	Nf	Ie	Nf	Ie	Nf	Ie	Nf	Ie
24	3	10,056	0	18,170	0	80,567	0	109,726	0	190,920	0
25	4	10,657	0	37,655	0	56,285	0	129,598	0	227,682	0
26	5	59,689	0	369,912	0	460,779	0	476,325	0	585,249	0
27	8	168,933	0	259,950	0	302,092	0	310,718	0	330,192	0
28	10	43,466	0	393,550	0	411,854	0	454,095	0	474,422	0
29	2	15,223	0	55,718	0	95,254	0	131,418	0	171,995	0
30	3	12,641	0	54,822	0	118,927	0	201,965	0	333,175	0
31	4	26,235	0	123,716	0	148,183	0	242,535	0	397,066	0
32	5	35,365	0	86,758	0	186,860	0	285,371	0	316,702	0
33	6	49,087	0	71,418	0	159,987	0	184,087	0	296,770	0
34	7	50,237	0	132,768	0	173,955	0	204,081	0	348,809	0
35	5	14,815	0	53,394	0	79,235	0	129,480	0	206,980	0
36	2	3,744	0	9,574	0	23,355	0	55,947	0	82,518	0
37	5	3,847	0	12,239	0	29,411	0	70,599	0	107,264	0

NPROB = problem number given in sect. 4.1.

Ie = 0 success (IOUT > 0) (claimed by SIGMA)

= 1 failure (IOUT ≤ 0) (claimed by SIGMA)

NSUC = see sect. 3.2.10.

Nf = total number of function evaluations including the ones needed to compute the "random" gradient

TABLE 2

UNIVAC

N_{SUC} =		1	2	3	4	5
NPROB	N					
1	1	1	1	1	1	1
2	1	1	1	1	1	1
3	1	1	1	1	1	1
4	2	1	1	1	1	1
5	2	1	1	1	1	1
6	2	1	1	1	1	1
7	2	1	1	1	1	1
8	2	3	1	1	1	1
9	2	3	3	1	1	1
10	2	1	1	1	1	1
11	2	1	1	1	1	1
12	2	1	1	1	1	1
13	2	1	1	1	1	1
14	2	1	1	1	1	1
15	2	1	1	1	1	1
16	2	1	1	1	1	1
17	2	1	1	1	1	1
18	4	3	3	1	1	1
19	4	3	1	1	1	1
20	4	3	1	1	1	1
21	5	1	1	1	1	1
22	6	1	1	1	1	1
23	7	1	1	1	1	1
24	5	1	1	1	1	1
25	4	1	1	1	1	1
26	5	1	1	1	1	1
27	8	3	3	3	1	1
28	10	1	1	1	1	1

Table 2 (continued)

UNIVAC (continued)

N_{SUC} =		1	2	3	4	5
NPROB	N					
29	2	1	1	1	1	1
30	3	3	1	1	1	1
31	4	3	1	1	1	1
32	5	1	1	1	1	1
33	6	1	1	1	1	
34	7	3	1	1	1	
35	5	1	1	1	1	1
36	2	3	3	3	3	3
37	5	3	3	3	3	

Table 2 (continued)

VAX

N_{SUC} =		1	2	3	4	5
NPROB	N					
1	1	1	1	1	1	1
2	1	1	1	1	1	1
3	1	1	1	1	1	1
4	2	1	1	1	1	1
5	2	1	1	1	1	1
6	2	1	1	1	1	1
7	2	1	1	1	1	1
8	2	3	3	3	3	1
9	2	1	1	1	1	1
10	2	1	1	1	1	1
11	2	1	1	1	1	1
12	2	1	1	1	1	1
13	2	1	1	1	1	1
14	2	1	1	1	1	1
15	2	1	1	1	1	1
16	2	1	1	1	1	1
17	2	1	1	1	1	1
18	4	3	1	1	1	1
19	4	1	1	1	1	1
20	4	1	1	1	1	1
21	5	3	1	1	1	1
22	6	1	1	1	1	1
23	2	1	1	1	1	1
24	3	1	1	1	1	1
25	4	1	1	1	1	1
26	5	1	1	1	1	1
27	8	1	1	1	1	1
28	10	1	1	1	1	1

Table 2 (continued)

VAX (continued)

N_{SUC} =		1	2	3	4	5
NPROB	N					
29	2	1	1	1	1	1
30	3	1	1	1	1	1
31	4	1	1	1	1	1
32	5	1	1	1	1	1
33	6	1	1	1	1	1
34	7	1	1	1	1	1
35	5	1	1	1	1	1
36	2	3	3	3	3	3
37	5	3	3	3	3	3

1 = success correctly claimed

2 = failure correctly claimed

3 = incorrect claim

4 = overflow

TABLE 3

		UNIVAC					VAX				
N_{SUC} =		1	2	3	4	5	1	2	3	4	5
Totals	1	26	32	34	35	35	32	34	34	34	35
	2	0	0	0	0	0	0	0	0	0	0
	3	11	5	3	2	2	5	3	3	3	2
	4	0	0	0	0	0	0	0	0	0	0

1 = success correctly claimed

2 = failure correctly claimed

3 = incorrect claim

4 = overflow

5. Conclusions.

The SIGMA package presented here seems to perform quite well on the proposed test problems.

As it is shown in [10] some of the test problems are very hard; for example, Problem 28 (N = 10) has a single global minimizer and a number of local minimizers of order 10^{10} in the region $|x_i| < 10$ $i = 1,2, \ldots, 10$.

Table 2 shows that from the point of view of the effectiveness measured by the number of correctly claimed successes the performance of SIGMA is very satisfactory; moreover, it is remarkably machine independent (note that completely different pseudo-random numbers sequences are generated by the algorithm on the two test machines). The results in Table 2 also suggest that the performance of SIGMA is very satisfactory from the point of view of dependability (only 2 incorrect claims on the "large" dynamic range machine when $N_{SUC} > 3$ and on the "small" dynamic range machine when $N_{SUC} > 4$) and robustness (no overflows on both machines).

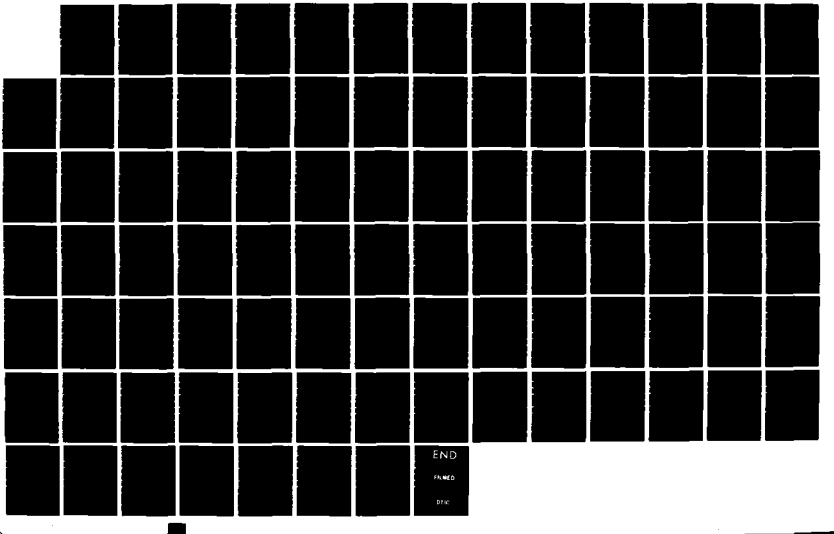
Unfortunately, given the state of the art on mathematical software for global optimization, it has not been possible to make conclusive comparisons with other packages.

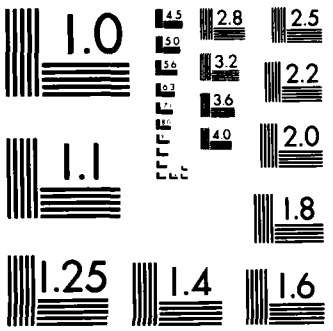
Finally, we note that a smaller value of N_{SUC} gives a much cheaper method (less function evaluations) at the expense of a loss in effectiveness (greater number of failures).

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, and Prof. A. Rinooy Kan for bringing to our attention ref. [9].

AD-A157 546 A NEW METHOD FOR GLOBAL OPTIMIZATION BASED ON STOCHASTIC DIFFERENTIAL EQUATIONS(U) CAMERINO UNIV (ITALY) MATHEMATICS INST F ALUFFI-PENTINI ET AL. 3/3

UNCLASSIFIED DEC 84 DAJA37-81-C-0740 F/G 12/1 NL





REFERENCES

[1] Aluffi-Pentini F., Parisi V., Zirilli F.: "Global optimization and stochastic differential equations," submitted to Journal of Optimization Theory and Applications.

[2] Zirilli F.: "The use of ordinary differential equations in the solution of nonlinear systems of equations," in "Nonlinear Optimization 1981," M.J.D. Powell, Editor, Academic Press, London, 1982, 39-47.

[3] Aluffi-Pentini F., Parisi V., Zirilli F.: "A differential equations algorithm for nonlinear equations," ACM Transactions on Mathematical Software, 10, (1984), 299-316.

[4] Aluffi-Pentini F., Parisi V., Zirilli F.: "Algorithm 617 DAFNE. A differential-equations algorithm for nonlinear equations," ACM Transactions on Mathematical Software, 10, (1984), 317-324.

[5] Powell M.J.D. (Editor): "Nonlinear Optimization 1981," Academic Press, London, 1982.

[6] Dennis J. E., Schnabel R. B.: "Numerical methods for unconstrained optimization and nonlinear equations," Prentice Hall, Inc., London, 1983.

[7] Schuss Z.: "Theory and applications of stochastic differential equations," J. Wiley & Sons, New York, 1980, chapter 8.

[8] Angeletti A., Castagnari C., Zirilli F.: "Asymptotic eigenvalue degeneracy for a class of one dimensional Fokker-Planck operators," to appear in J. of Math. Phys.

[9] Kirkpatric S., Gelatt C.D. Jr., Vecchi M.P.: "Optimizat... simulated annealing," Science 220 , (1983), 671-680.

[10] Aluffi-Pentini F., Parisi V., Zirilli F.: "Test problem... global optimization software," submitted to A.C.M. Transact... on Mathematical Software.

APPENDIX A5

Algorithm SIGMA. A stochastic-integration global minimization algorithm

by F. Aluffi-Pentini, V. Parisi, F. Zirilli

(submitted to ACM Transactions on Mathematical Software).

ALGORITHM ...

SIGMA — A Stochastic-Integration Global Minimization Algorithm

Filippo Aluffi-Pentini
Bari University

Valerio Parisi
2nd Rome University

Francesco Zirilli
Salerno University

Categories and Subject Descriptors:

G.1.6 [Numerical Analysis]: Optimization – ;
G.4 [Mathematical Software]: – ;

General Terms: Algorithms

Additional Key Words and Phrases: Global Optimization, Stochastic Differential Equations

Language: FORTRAN

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 n. DAJA-37-81-C-0740 with the University of Camerino, Italy.

Authors' addresses:

F. Aluffi-Pentini, Dipartimento di Matematica, Università di Bari, 70125 Bari (Italy);
V. Parisi, Istituto di Fisica, 2ª Università di Roma "Tor Vergata", Via Orazio Raimondo, 00173 (La Romanina) Roma (Italy);
F. Zirilli, Istituto di Matematica, Università di Salerno, 84100 Salerno (Italy).

1. PURPOSE

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 $x_1, \ldots, x_N$.

2. 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

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

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

where h_k is the time integration steplength, $\frac{1}{N}\tilde{\underline{Y}}(\underline{\xi}_k)$ 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.

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 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 values for ε_p in the second continuation trajectory after branching, and trial-start values 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}.

3. 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.

3.1. 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.

3.2. 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 4.

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. 4.5.1) 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. 4.5.2) 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. 4.4).

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

|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 ($1 \leq KSUC < NSUC$)

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.

4.5.2. 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. 4.2).

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

The subroutine PTRIAL is called (if IPRINT $\geq$ 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

4.5. 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).

4.5.1. 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

4.4 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 $0 < NPMAX\emptyset < 150$)

NTRIAL (say $NTRIAL > 50$ and $NTRIAL > 5 \cdot NSUC$)

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. 4.4).

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. 4.4).

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. 4.5.2.).

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. 4.5.2) and no lower value for f_{OPT} was found in the following trials.

4.3. 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. 4.5.2.

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. 2) 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. 4.5.2. The parameters are

KP, NF, XOPT, FOPT

XPFMIN, FPFMIN, FPFMAX, FTFMIN, FTFMAX, FTFOPT

ISTOP, ISTOPT, KSUC

and are described in sect. 4.5.2.

4.2. 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

4. 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. 4.1, the parameters of the CALL statement are described in sect. 4.2. Some guidelines for the choice of the values of the input parameters are given in sect. 4.3. A sample driver subroutine (SIGMA1) which calls SIGMA is described in sect. 4.4: 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. 4.5.

4.1. 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

(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.

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 4.5.2)

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

4.6. 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.

4.7. 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 evaluation. The printout showed that if N_{SUC} had been equal to 1 (resp. 2), the minimum would have been found with only 2697 (resp. 8543) 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.

```
      C
      C  MAIN PROGRAM (SAMPLE VERSION)
      C  CALLS  SIGMA  VIA THE DRIVER SUBROUTINE  SIGMA1
      C
            DOUBLE PRECISION XC,XMIN,FMIN
      C
            DIMENSION XC(2),XMIN(2)
      C
      C TEST PROBLEM DATA
      C
      C PROBLEM DIMENSION
            N = 2
      C
      C INITIAL POINT
            XC(1) = 0.D0
            XC(2) = 0.D0
      C
      C SET INPUT PARAMETERS
            NSUC = 2
            IPRINT = 0
      C
      C CALL DRIVER SUBROUTINE  SIGMA1
            CALL SIGMA1(N,XC,NSUC,IPRINT,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.D0-2.1D0)*XX+4.D0)*XX+X(1)*X(2)
           *     +4.D0*(YY-1.D0)*YY
            RETURN
            END
```

Fig. 1 -- List of the Sample Program

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 Technical 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.

APPENDIX A6

The FORTRAN package SIGMA.

```
@FTN,S ACM/A.FILESIGMA,TPFS.FILESIGMA
FTN 10R1A  02/12/85-22:02(0,)
      1.          SUBROUTINE SIGMA1(N,X0,NSUC,IPRINT,XMIN,FMIN,NFEV,IOUT)
      2.    C
      3.    C  SIGMA1  IS A 'DRIVER' SUBROUTINE WHICH SIMPLY CALLS THE PRINCIPAL
      4.    C  SUBROUTINE  SIGMA  AFTER HAVING ASSIGNED DEFAULT VALUES TO A NUMBER
      5.    C  OF INPUT PARAMETERS OF  SIGMA , AND WAS THEREFORE A CONSIDERABLE
      6.    C  LOWER NUMBER OF INPUT PARAMETERS.
      7.    C  IT CAN BE USED AS A SIMPLE EXAMPLE OF HOW TO CALL  SIGMA , BUT
      8.    C  ALSO AS AN EASY-TO-USE DRIVER FOR THE AVERAGE USER, WHICH MAY FIND
      9.    C  IT EASIER TO CALL  SIGMA1  INSTEAD OF  SIGMA , THUS AVOIDING THE
     10.    C  TROUBLE OF ASSIGNING A VALUE TO ALL THE INPUT PARAMETERS OF  SIGMA .
     11.    C  ALL THE PARAMETERS IN THE DEFINITION OF  SIGMA1  HAVE THE SAME MEA-
     12.    C  NING AS IN  SIGMA .
     13.    C
     14.    C  THE USER OF  SIGMA1  MUST ONLY GIVE VALUES TO THE INPUT PARAMETERS
     15.    C       N, X0, NSUC, IPRINT
     16.    C  AND OBTAINS ON OUTPUT THE SAME OUTPUT PARAMETERS OF  SIGMA
     17.    C       XMIN, FMIN, NFEV, IOUT
     18.    C
     19.    C  WE RECALL HERE THE MEANING OF THE ABOVE PARAMETERS
     20.    C
     21.    C  N          IS THE PROBLEM DIMENSION (NUMBER OF VARIABLES)
     22.    C  X0         IS AN  N-VECTOR CONTAINING THE INITIAL VALUES OF THE
     23.    C             X-VARIABLES
     24.    C  NSUC       IS THE NUMBER OF SUCCESSFUL TRIALS (WITH THE SAME FINAL
     25.    C             VALUE  FOPT ) AFTER WHICH THE COMPUTATION IS STOPPED.
     26.    C  IPRINT     IS AN INDEX USED TO CONTROL THE AMOUNT OF PRINTED OUTPUT
     27.    C             BY CONTROLLING THE CALLS TO THE USER-SUPPLIED OUTPUT SUB-
     28.    C             ROUTINES  PTSEG  (END-OF-SEGMENT OUTPUT),  PTRIAL  (END-
     29.    C             OF-TRIAL OUTPUT), AND  PTKSUC (END-OF-TRIAL OUTPUT RELATED
     30.    C             TO THE COUNT OF SUCCESSFUL TRIALS), WHICH ARE DESCRIBED
     31.    C             BELOW.
     32.    C             IPRINT.LT.0   NO CALL TO THE OUTPUT SUBROUTINES
     33.    C             IPRINT.EQ.0   CALL ONLY  PTRIAL  AND  PTKSUC
     34.    C             IPRINT.GT.0   CALL ALL OUTPUT SUBROUTINES.
     35.    C  XMIN       IS AN N-VECTOR CONTAINING THE COORDINATES OF THE POINT
     36.    C             (OR POSSIBLY ONE OF THE POINTS) WHERE THE FINAL VALUE  FMIN
     37.    C             OF  FOPT  WAS FOUND.
     38.    C  FMIN       IS THE FINAL VALUE OF THE BEST CURRENT MINIMUM FUNCTION
     39.    C             VALUE  FOPT.
     40.    C  NFEV       IS THE TOTAL NUMBER OF FUNCTION EVALUATION (INCLUDING
     41.    C             THOSE USED FOR THE COMPUTATION OF DERIVATIVES, AND FOR
     42.    C             THE REJECTED TIME-INTEGRATION STEPS).
     43.    C  IOUT       IS THE INDICATOR OF THE STOPPING CONDITIONS, AS FOLLOWS
     44.    C             IF  IOUT = -99  A FATAL ERROR WAS DETECTED WHEN PERFOR-
     45.    C                MING SOME PRELIMINARY CHECKING OF THE INPUT DATA, AND
     46.    C                THE ALGORITHM WAS NOT EVEN STARTED
     47.    C             OTHERWISE THE ALGORITHM WAS STARTED, AND THE VALUE OF
     48.    C             IOUT  IS THE FINAL VALUE OF THE INTERNAL PARAMETER  ISTOPT
     49.    C             (AN OUTPUT INDICATOR OF THE USER-SUPPLIED SUBROUTINE
     50.    C             PTRIAL ).
     51.    C             SUCCESS IS CLAIMED BY THE ALGORITHM IF  IOUT .GT. 0,
     52.    C             I.E. IF AT LEAST ONE OF THE TRIALS STOPPED UNIFORMLY AT THE
     53.    C             LEVEL OF THE CURRENT  FOPT.
     54.    C
     55.          DOUBLE PRECISION X0,XMIN,FMIN
```

```
      DOUBLE PRECISION DX,EPS,H,TOLABS,TOLREL
      DOUBLE PRECISION VRMAX,VRMIN,XRMAX,XRMIN
      DIMENSION XO(N),XMIN(N)
      DIMENSION XRMIN(100),XRMAX(100)
      DATA VRMIN,VRMAX /-1.D4,1.D4/
      DATA NTRILD/50/
C
      H = 1.D-10
      EPS = 1.D0
      DX = 1.D-9
      IRAND = 0
      NTRAJ = 0
      ISEGBR = 0
      INKPBR = 0
      KPBRO = 0
      NPMIN = 10
      NPMAXO = 100
      INPMAX = 50
      NTRIAL = MAXO(NTRILD,5*NSUC)
      TOLREL = 1.D-3
      TOLABS = 1.D-6
      KPASCA = 10
      IF(N.GT.5)KPASCA = 300
      INHP = 1
      DO 1 IX = 1,N
      XRMIN(IX)=VRMIN
      XRMAX(IX)=VRMAX
    1 CONTINUE
C
      CALL SIGMA ( N, XO, H, EPS, DX,
     1             NTRAJ, ISEGBR, KPBRO, INKPBR,
     2             NPMIN, NPMAXO, INPMAX,
     3             NSUC, NTRIAL, TOLREL, TOLABS, XRMIN, XRMAX,
     4             KPASCA, IRAND, INHP, IPRINT,
     5             XMIN, FMIN, NFEV, IOUT )
C
      RETURN
      END
```

```
      SUBROUTINE SIGMA ( N, XO, H, EPS, DX,
     1                   NTRAJ, ISEGBR, KPBRO, INKPBR,
     2                   NPMIN, NPMAXO, INPMAX,
     3                   NSUC, NTRIAL, TOLREL, TOLABS, XRMIN, XRMAX,
     4                   KPASCA, IRAND, INHP, IPRINT,
     5                   XMIN, FMIN, NFEV, IOUT )
C
C  THE SUBROUTINE  SIGMA  IS THE PRINCIPAL SUBROUTINE OF THE PACKAGE
C  SIGMA, WHICH ATTEMPTS TO FIND A GLOBAL MINIMIZER OF A REAL VALUED
C  FUNCTION  F(X) = F(X1,...,XN)  OF  N  REAL VARIABLES  X1,...,XN.
C  THE ALGORITHM AND THE PACKAGE ARE DESCRIBED IN DETAIL IN THE TWO
```

```
C  PAPERS PUBLISHED IN THE SAME ISSUE OF THE A.C.M. TRANSACTIONS ON
C  MATHEMATICAL SOFTWARE, BOTH BY
C     F. ALUFFI-PENTINI, V. PARISI, F. ZIRILLI.
C  (1)  A GLOBAL MINIMIZATION ALGORITHM USING STOCHASTIC DIFFERENTIAL
C       EQUATIONS
C  (2)  ALGORITHM SIGMA, A STOCHASTIC-INTEGRATION GLOBAL MINIMIZATION
C       ALGORITHM.
C  THE SOFTWARE IMPLEMENTATION AND ITS USAGE ARE DESCRIBED IN (2).
C
C  METHOD
C
C  A GLOBAL MINIMIZER OF  F(X)  IS SOUGHT BY MONITORING THE VALUES OF
C  F  ALONG TRAJECTORIES GENERATED BY A SUITABLE (STOCHASTIC) DISCRE-
C  TIZATION OF A FIRST-ORDER STOCHASTIC DIFFERENTIAL EQUATION INSPIRED
C  BY STATISTICAL MECHANICS. STARTING FROM AN INITIAL POINT  X0 ,
C  X  IS UPDATED BY THE (STOCHASTIC) DISCRETIZATION STEP
C        X = X + DX1 + DX2
C  WHERE  DX1 = - H * GAM              (FIRST HALF-STEP)
C         DX2 = EPS * SQRT(H) * U      (SECOND HALF-STEP)
C  AND  H  IS THE TIME-INTEGRATION STEPLENGTH,
C  GAM/N  IS COMPUTED AS A FINITE-DIFFERENCE APPROXIMATION TO THE
C     DIRECTIONAL DERIVATIVE OF  F  ALONG AN ISOTROPICALLY RANDOM
C     DIRECTION,
C  EPS  IS A POSITIVE 'NOISE' COEFFICIENT, AND
C  U  IS A RANDOM SAMPLE FROM AN N-DIMENSIONAL GAUSSIAN DISTRIBUTION.
C  WE CONSIDER THE SIMULTANEOUS EVOLUTION OF A GIVEN FIXED NUMBER
C  NTRAJ  OF TRAJECTORIES DURING AN OBSERVATION PERIOD IN WHICH FOR
C  EACH TRAJECTORY  EPS  IS FIXED WHILE  H  AND THE SPATIAL DISCRETI-
C  ZATION INCREMENT  DX  FOR COMPUTING  GAM  ARE AUTOMATICALLY
C  ADJUSTED BY THE ALGORITHM.
C  AFTER EVERY OBSERVATION PERIOD ONE OF THE TRAJECTORIES IS DISCARDED,
C  ALL OTHER TRAJECTORIES CONTINUE UNPERTURBED, AND ONE OF THEM IS SE-
C  LECTED FOR BRANCHING, I.E. GENERATING ALSO A SECOND PERTURBED CONTI-
C  NUATION, WITH DIFFERENT STARTING  EPS  AND  DX  (AND THE SAME
C  'PAST HISTORY' OF THE FIRST).
C  THE SET OF SIMULTANEOUS TRAJECTORIES IS CONSIDERED A SINGLE TRIAL,
C  AND THE COMPLETE ALGORITHM IS A SET OF REPEATED TRIALS.
C  A TRIAL IS STOPPED, AT THE END OF AN OBSERVATION PERIOD, AND AFTER
C  HAVING DISCARDED THE WORST TRAJECTORY, IF ALL THE FINAL VALUES OF
C  F  FOR THE REMAINING TRAJECTORIES ARE EQUAL (WITHIN NUMERICAL TO-
C  LERANCES, AND POSSIBLY AT DIFFERENT POINTS  X) TO THEIR MINIMUM
C  VALUE  FTFMIN ('UNIFORM STOP AT THE LEVEL  FTFMIN ').
C  AN UNIFORM STOP IS CONSIDERED SUCCESSFUL ONLY IF THE FINAL VALUE
C  FTFMIN  IS (NUMERICALLY) EQUAL TO THE CURRENT BEST MINIMUM  FOPT
C  FOUND SO FAR FROM ALGORITHM START.
C  A TRIAL IS ALSO ANYWAY STOPPED (UNSUCCESSFULLY) IF A GIVEN MAXIMUM
C  NUMBER  NPMAX  OF OBSERVATION PERIODS HAS ELAPSED.
C  TRIALS ARE REPEATED WITH DIFFERENT OPERATING CONDITIONS (INITIAL
C  POINT, MAX TRIAL LENGTH  NPMAX , SEED OF NOISE GENERATOR, POLICY
C  FOR CHOOSING THE STARTING  EPS  FOR THE PERTURBED CONTINUATION,
C  AND TRIAL-START VALUE OF  EPS ).
C  THE ALGORITHM IS STOPPED, AT THE END OF A TRIAL, IF A GIVEN NUMBER
C  NSUC  OF UNIFORM STOPS AT THE CURRENT  FOPT  LEVEL HAS BEEN REACHED,
C  OR ANYWAY IF A GIVEN MAXIMUM NUMBER  NTRIAL  OF TRIALS HAS BEEN
C  REACHED.
C  SUCCESS IS CLAIMED IF AT LEAST ONE UNIFORM STOP OCCURRED AT THE
C  FINAL VALUE OF  FOPT.
```

```
C
C  CALL STATEMENT
C
C  THE CALL STATEMENT IS
C     CALL SIGMA ( N, XO, H, EPS, DX,
C                  NTRAJ, ISEGBR, KPBRO, INKPBR,
C                  NPMIN, NPMAXO, INPMAX,
C                  NSUC, NTRIAL, TOLREL, TOLABS, XRMIN, XRMAX,
C                  KPASCA, IRAND, INHP, IPRINT,
C                  XMIN, FMIN, NFEV, IOUT )
C
C  CALL PARAMETERS
C
C  INPUT PARAMETERS ARE THOSE IN LINES 1,3,4,5 OF THE CALL STATEMENT,
C  INPUT-OUTPUT PARAMETERS ARE THOSE IN LINE 2,
C  OUTPUT PARAMETERS ARE THOSE IN LINE 6.
C  NOTE THAT A NUMBER OF OTHER (INTERNAL) PARAMETERS CAN BE OBTAINED
C  BY MEANS OF THE USER-SUPPLIED OUTPUT SUBROUTINES  PTSEG, PTRIAL,
C  AND  PTKSUC, WHICH ARE DESCRIBED BELOW.
C
C  DESCRIPTION OF THE CALL PARAMETERS
C
C  N         IS THE PROBLEM DIMENSION (NUMBER OF VARIABLES)
C  XO        IS AN  N-VECTOR CONTAINING THE INITIAL VALUES OF THE
C            X-VARIABLES
C  H         IS THE INITIAL VALUE OF THE TIME-INTEGRATION STEPLENGTH.
C  EPS       IS THE INITIAL VALUE OF THE NOISE COEFFICIENT.
C  DX        IS THE INITIAL VALUE OF THE MAGNITUDE OF THE DISCRETIZATION
C            INCREMENT FOR COMPUTING THE FINITE-DIFFERENCE DERIVATIVES.
C  NTRAJ     IS THE NUMBER OF SIMULTANEOUS TRAJECTORIES.
C            (NOTE HOWEVER THAT IF THE INPUT VALUE IS ZERO,  NTRAJ  IS
C            SET TO A DEFAULT VALUE (NTRAJ = 7), AND IF THE INPUT VALUE
C            IS OTHERWISE OUTSIDE THE INTERVAL (3,20) NTRAJ IS SET TO
C            THE NEAREST EXTREME VALUE).
C  ISEGBR, KPBRO, INKPBR  DETERMINE, AT THE END OF AN OBSERVATION
C            PERIOD, WHICH ONE OF THE SIMULTANEOUS TRAJECTORIES
C            IS TO BE BRANCHED, AS FOLLOWS.
C            BRANCHING IS NORMALLY PERFORMED ON THE TRAJECTORY WHICH
C            OCCUPIES THE PLACE  ISEGBR  IN THE TRAJECTORY SELECTION OR-
C            DERING, EXCEPT AT (THE END OF) EXCEPTIONAL OBSERVATION
C            PERIODS, WHERE THE FIRST TRAJECTORY IN THE ORDERING IS
C            BRANCHED. EXCEPTIONAL BRANCHING OCCURS AT THE OBSERVATION
C            PERIODS NUMBERED  KP = KPBRO + J*INKPBR, (J = 1,2,3,...).
C            THEREFORE  ISEGBR  SELECTS THE LEVEL (IN THE ORDERING) AT
C            WHICH NORMAL BRANCHING OCCURS, WHILE  KPBRO  AND  INKPBR
C            SELECT THE FIRST OCCURRENCE AND THE REPETITION FREQUENCY
C            OF THE EXCEPTIONAL OBSERVATION PERIODS.
C            (NOTE HOWEVER THAT IF ONE OF THE INPUT VALUES IS ZERO,
C            THE CORRESPONDING VARIABLE IS SET TO A DEFAULT VALUE
C            ISEGBR = INT((1+NTRAJ)/2), INKPBR = 10, KPBRO = 3.
C            IF THE INPUT VALUE FOR  ISEGBR  IS OTHERWISE OUTSIDE THE
C            INTERVAL  (1,NTRAJ),  ISEGBR  IS SET TO THE NEAREST
C            EXTREME VALUE, AND IF KPBRO  HAS A VALUE NOT INSIDE THE
C            INTERVAL  (1,INKPBR), IT IS ASSIGNED THE SAME VALUE
C            MODULO  INKPBR).
C  NPMIN     IS THE MINIMUM DURATION OF A TRIAL, I.E. THE MINIMUM
C            NUMBER OF OBSERVATION PERIODS THAT SHOULD ELAPSE BEFORE
```

```
C          STARTING TO CHECK THE TRIAL STOPPING CRITERIA.
C  NPMAX0  IS THE MAXIMUM DURATION OF THE FIRST TRIAL, I.E. THE
C          VALUE, FOR THE FIRST TRIAL, OF MAXIMUM ACCEPTABLE
C          NUMBER  NPMAX  OF OBSERVATION PERIODS IN A TRIAL.
C  INPMAX  IS THE INCREMENT FOR  NPMAX , WHEN  NPMAX  IS VARIED
C          FROM ONE TRIAL TO THE FOLLOWING ONE.
C  NSUC    IS THE NUMBER OF SUCCESSFUL TRIALS (WITH THE SAME FINAL
C          VALUE  FOPT ) AFTER WHICH THE COMPUTATION IS STOPPED.
C  TOLREL  AND  TOLABS  ARE THE RELATIVE AND ABSOLUTE TOLERANCES
C          FOR STOPPING A SINGLE TRIAL.
C  XRMIN,  XRMAX  ARE N-VECTORS DEFINING THE ADMISIBLE REGION FOR
C          THE X-VALUES, WITHIN WHICH THE FUNCTION VALUES CAN BE
C          SAFELY COMPUTED.
C  KPASCA  IS THE MINIMUM NUMBER OF TRAJECTORY SEGMENTS THAT SHOULD
C          ELAPSE BEFORE THE RESCALING PROCEDURES ARE ACTIVATED.
C  IRAND   IS A CONTROL INDEX FOR THE INITIALIZATION OF THE RANDOM
C          NUMBER GENERATOR.
C          IRAND.GT.0     THE GENERATOR IS INITIALIZED, BEFORE STAR-
C                         TING THE TRIAL  KT, WITH SEED  IRAND+KT-1
C          IRAND.LE.0     THE GENERATOR IS INITIALIZED (WITH SEED 0)
C                         ONLY AT THE FIRST CALL OF SIGMA
C  INHP    IS A CONTROL INDEX FOR SELECTING THE NUMBER  NHP  OF TIME-
C          INTEGRATION STEPS FOR OBSERVATION PERIOD  KP (DURATION OF
C          TRIAL  KP) AS FOLLOWS (LOG IS BASE 2)
C          INHP=1  NHP = 1 + INT(LOG(KP))       ('SHORT' DURATION)
C          INHP=2  NHP = INT(SQRT(KP))          ('MEDIUM' DURATION)
C          INHP=3  NHP = KP                     ('LONG' DURATION)
C  IPRINT  IS AN INDEX USED TO CONTROL THE AMOUNT OF PRINTED OUTPUT
C          BY CONTROLLING THE CALLS TO THE USER-SUPPLIED OUTPUT SUB-
C          ROUTINES  PTSEG (END-OF-SEGMENT OUTPUT),  PTRIAL  (END-
C          OF-TRIAL OUTPUT), AND  PTKSUC (END-OF-TRIAL OUTPUT RELATED
C          TO THE COUNT OF SUCCESSFUL TRIALS), WHICH ARE DESCRIBED
C          BELOW.
C          IPRINT.LT.0    NO CALL TO THE OUTPUT SUBROUTINES
C          IPRINT.EQ.0    CALL ONLY  PTRIAL  AND  PTKSUC
C          IPRINT.GT.0    CALL ALL OUTPUT SUBROUTINES.
C  XMIN    IS AN N-VECTOR CONTAINING THE COORDINATES OF THE POINT
C          (OR POSSIBLY ONE OF THE POINTS) WHERE THE FINAL VALUE  FMIN
C          OF  FOPT  WAS FOUND.
C  FMIN    IS THE FINAL VALUE OF THE BEST CURRENT MINIMUM FUNCTION
C          VALUE  FOPT.
C  NFEV    IS THE TOTAL NUMBER OF FUNCTION EVALUATION (INCLUDING
C          THOSE USED FOR THE COMPUTATION OF DERIVATIVES, AND FOR
C          THE REJECTED TIME-INTEGRATION STEPS).
C  IOUT    IS THE INDICATOR OF THE STOPPING CONDITIONS, AS FOLLOWS
C          IF  IOUT = -99  A FATAL ERROR WAS DETECTED WHEN PERFOR-
C             MING SOME PRELIMINARY CHECKING OF THE INPUT DATA, AND
C             THE ALGORITHM WAS NOT EVEN STARTED
C          OTHERWISE THE ALGORITHM WAS STARTED, AND THE VALUE OF
C          IOUT  IS THE FINAL VALUE OF THE INTERNAL PARAMETER  ISTOPT
C          (AN OUTPUT INDICATOR OF THE USER-SUPPLIED SUBROUTINE
C          PTRIAL , DESCRIBED BELOW).
C          SUCCESS IS CLAIMED BY THE ALGORITHM IF  IOUT .GT. 0,
C          I.E. IF AT LEAST ONE OF THE TRIALS STOPPED WITH A POSITIVE
C          VALUE OF THE TRIAL STOPPING INDICATOR  ISTOP (AN OUTPUT
C          INDICATOR OF THE USER-SUPPLIED SUBROUTINE  PTRIAL ,
C          DESCRIBED BELOW), AND NO LOWER VALUE FOR  FOPT  WAS FOUND
```

```
C           IN THE FOLLOWING TRIALS.
C
C  USER-SUPPLIED SUBPROGRAMS
C
C  THE USER MUST PROVIDE THE FUNCTION  FUNCT  TO COMPUTE  F(X),
C  AND THE THREE OUTPUT SUBROUTINE  PTSEG, PTRIAL, PTRSUC .
C  THE CALLS TO THE OUTPUT SUBROUTINES ARE CONTROLLED BY  IPRINT
C  (INPUT PARAMETER TO  SIGMA).
C  A USER NOT INTERESTED IN USING ANY ONE OF THE OUTPUT SUBROUTINES
C  MUST PROVIDE A DUMMY SUBROUTINE (WITH  RETURN  AS THE ONLY
C  EXECUTABLE STATEMENT) TO AVOID UNRESOLVED REFERENCES.
C  IN THE FOLLOWING DESCRIPTION ALL NON-INTEGER ARGUMENTS ARE
C  DOUBLE PRECISION (INTEGER ARGUMENTS ARE INDICATED BY MEANS OF THE
C  FORTRAN IMPLICIT TYPE DEFINITION CONVENTION).
C
C     THE FUNCTION  FUNCT
C
C     FUNCT  MUST RETURN AS ITS VALUE THE VALUE AT  X  OF THE FUNCTION
C     TO BE MINIMIZED
C     THE DEFINITION STATEMENT IS
C        DOUBLE PRECISION FUNCTION  FUNCT (N, X)
C     WHERE
C     N    IS THE (INPUT) DIMENSION OF THE PROBLEM.
C     X    IS THE (INPUT) N-VECTOR CONTAINING THE COORDINATES OF THE
C          POINT  X  WHERE THE FUNCTION IS TO BE COMPUTED.
C
C     THE SUBROUTINE  PTSEG
C
C     PTSEG  IS CALLED (IF  IPRINT.GT.0) AT THE END OF EVERY OBSER-
C     VATION PERIOD.
C     THE DEFINITION STATEMENT IS
C          SUBROUTINE  PTSEG ( N, XPFMIN, FPFMIN, FPFMAX,
C                                              KP, NFEV, IPRINT )
C     WHERE
C     N    IS THE (INPUT) DIMENSION OF THE PROBLEM
C     FPFMIN, FPFMAX  ARE RESPECTIVELY THE MINIMUM AND THE MAXIMUM
C        AMONG THE VALUES OF  F(X) OBTAINED AT THE FINAL POINTS OF
C        THE TRAJECTORY SEGMENTS OF THE (JUST ELAPSED) OBSERVATION
C        PERIOD  KP.
C     XPFMIN   IS AN N-VECTOR CONTAINING THE COORDINATES OF THE
C        (FINAL) POINT (OR POSSIBLY ONE OF THE POINTS) WHERE
C        FPFMIN WAS OBTAINED.
C     KP   IS THE TOTAL NUMBER OF ELAPSED OBSERVATION PERIODS IN
C        THE CURRENT TRIAL.
C     NFEV   IS THE TOTAL NUMBER OF FUNCTION EVALUATIONS PERFORMED
C        FROM ALGORITHM START.
C
C     THE SUBROUTINE  PTRIAL
C
C     PTRIAL  IS CALLED (IF IPRINT.GE.0) AT THE END OF EVERY TRIAL.
C     THE DEFINITION STATEMENT IS
C          SUBROUTINE PTRIAL ( N, XOPT, FOPT,
C                              FTFMIN, FTFMAX, FTFOPT,
C                              ISTOP, ISTOPT, NFEV, KR, IPRINT )
C     WHERE
C     N    IS THE (INPUT) DIMENSION OF THE PROBLEM.
C     XOPT   IS AN N-VECTOR CONTINING THE COORDINATES OF THE
```

```
C        POINT (OR POSSIBLY ONE OF THE POINTS) WHERE THE CURRENT
C        MINIMUM  FOPT  WAS OBTAINED.
C     FOPT   IS THE CURRENT BEST MINIMUM VALUE FOUND FOR  F  FROM
C        ALGORITHM START ( FOPT  IS UPDATED WHENEVER A FUNCTION
C        VALUE IS COMPUTED).
C     FTFMIN, FTFMAX   ARE RESPECTIVELY THE MINIMUM AND THE MAXIMUM
C        AMONG THE VALUES OF  F(X)  OBTAINED AT THE FINAL POINTS OF
C        THE LAST TRAJECTORY SEGMENTS OF THE CURRENT TRIAL.
C     FTFOPT   IS THE CURRENT MINIMUM VALUE OF  FTFMIN  AMONG THE
C        TRIALS WHICH DID NOT STOP FOR REACHING THE MAXIMUM ALLOWED
C        NUMBER OF SEGMENTS (STOPPING INDICATOR  ISTOP = 0, SEE
C        BELOW).  FTFOPT  IS USED BY  SIGMA  TO COMPUTE  NSUC  (IN OUT
C        PARAMETER TO THE OUTPUT SUBROUTINE  PTKSUK , SEE BELOW).
C     KP   IS THE TOTAL NUMBER OF ELAPSED OBSERVATION PERIODS IN
C        THE CURRENT TRIAL.
C     NFEV   IS THE TOTAL NUMBER OF FUNCTION EVALUATIONS PERFORMED
C        FROM ALGORITHM START.
C     ISTOP   IS THE INDICATOR OF THE STOPPING CONDITION OF THE TRIAL,
C        AS FOLLOWS
C        ISTOP = 0
C           THE MAXIMUM NUMBER  NPMAX  OF OBSERVATION PERIODS HAS
C           BEEN REACHED.
C        ISTOP.NE.0
C           ALL THE END-OF-SEGMENT VALUES OF  F(X) , (EXCEPT FOR THE
C           JUST DISCARDED SEGMENT) ARE CLOSE ENOUGH TO THEIR COMMON
C           MINIMUM VALUE  FPFMIN , WITH RESPECT TO AN ABSOLUTE OR
C           RELATIVE DIFFERENCE CRITERION, TO BE CONSIDERED NUMERI-
C           CALLY EQUAL.
C           THE ABSOLUTE VALUE AND THE SIGN OF  ISTOP  HAVE THE
C           FOLLOWING MEANING.
C              THE ABSOLUTE VALUE INDICATES WHICH DIFFERENCE
C                 CRITERION WAS SATISFIED
C                 1   RELATIVE DIFFERENCE CRITERION SATISFIED
C                 2   ABSOLUTE DIFFERENCE CRITERION SATISFIED
C                 3   BOTH CRITERIA SATISFIED
C              THE SIGN OF  ISTOP  INDICATES THE RELATIONSHIP BETWEEN
C                 THE END-OF-TRIAL VALUE  FPFMIN  AND THE CURRENT
C                 BEST MINIMUM VALUE  FOPT  (WHICH IS UPDATED WHEN-
C                 EVER A FUNCTION  VALUE IS COMPUTED
C                 ISTOP.GT.0
C                    FPFMIN  IS NUMERICALLY EQUAL (W.R.T. AT LEAST
C                    ONE OF THE ABOVE  DIFFERENCE CRITERIA) TO  FOPT
C                 ISTOP.LT.0
C                    FPFMIN  IS NOT EVEN NUMERICALLY EQUAL TO  FOPT
C                    (AND THEREFORE CANNOT BE CONSIDERED AS AN
C                    ACCEPTABLE GLOBAL MINIMUM).
C     ISTOPT   IS THE VALUE OF THE TRIAL STOPPING INDICATOR  ISTOP
C        CORRESPONDING TO THE (CURRENT OR PAST) TRIAL WHERE  FTFOPT
C        WAS OBTAINED, WITH THE SIGN WHICH IS UPDATED ACCORDING TO
C        THE COMPARISON BETWEEN  FTFOPT  AND THE PRESENT VALUE OF
C        FOPT , AS DESCRIBED ABOVE.
C        THE FINAL VALUE OF  ISTOP IS RETURNED BY  SIGMA  AS THE VALUE
C        OF THE OUTPUT INDICATOR IOUT OF THE ALGORITHM STOPPING CON-
C        DITIONS (WHENEVER THE ALGORITHM WAS STARTED, IOUT.NE.-99,
C        SEE ABOVE).
C
C     THE SUBROUTINE  PTKSUC
```

```
C
C     PTKSUC  IS CALLED ONLY AT THE END OF EVERY SUCCESSFUL TRIAL
C     SUCH THAT AN INCREMENT OCCURRED IN THE VALUE  KSUC  OF THE
C     MAXIMUM NUMBER OF SUCCESSFUL TRIALS AT THE SAME (CURRENT OR
C     PAST) VALUE OF  FOPT .  A CALL TO  PTKSUC  THEREFORE PROVIDES
C     THE USER WITH THE OPERATIONALLY INTERESTING INFORMATION THAT
C     A FINAL SUCCESS CLAIM WOULD HAVE TAKEN PLACE, IF  NSUC  (INPUT
C     PARAMETER TO  SIGMA ) HAD BEEN GIVEN A LOWER VALUE, EQUAL TO
C     THE CURRENT  KSUC .
C     PTKSUC  IS CALLED ONLY IF  IPRINT.GE.0  AND  KSUC.LT.NSUC  .
C     THE DEFINITION STATEMENT IS
C         SUBROUTINE  PTKSUC ( KSUC )
C     WHERE  KSUC  IS THE INTEGER VARIABLE (1 .LE. KSUC .LE. NSUC)
C     DEFINED ABOVE.
C
      DOUBLE PRECISION X0,H,EPS,DX,TOLREL,TOLABS
      DOUBLE PRECISION XRMIN,XRMAX,XMIN,FMIN
      DOUBLE PRECISION EPSAG,EPSAP,EPSC
      DOUBLE PRECISION ERSMAX,EPSR,F,FOPT,FTFMAX
      DOUBLE PRECISION FTFMIN,FTFOPT
C
      DOUBLE PRECISION X,HC,DXC,VMVT,EPSCO,VMCOR,VCOR
      DOUBLE PRECISION XRMIC,XRMAC,XOPT,FOPTC
C
      DOUBLE PRECISION DIST,BIAS,GRAGRA,GRA
C
      DIMENSION X0(N),XMIN(N),XRMIN(N),XRMAX(N)
C
      COMMON /DINCOM/ X(100,20),HC(20),DXC(20),VMVT(20,19),EPSCO(20),
     1    VMCOR(20),VCOR(20),XRMIC(100),XRMAC(100),XOPT(100),FOPTC,
     2    IE(20),ISVT(20,19),KGEN,KTIM,NDIM,NTRAJC,NTRAJR,
     3    ISEGBC,INKPBC,KPBROC,NCF,IFEPC,INHPC
C
      COMMON /SCALE/ DIST(10,10,20),BIAS(10,20),GRAGRA(10,10,20),
     1    GRA(10,20),NGRA(20),LSCA,IDSCA,NX,NORD
C
C   DATA FOR THE VARIATION OF NOISE COEFFICIENT
C
      DATA EPSR/1.D4/,EPSAP/10.D0/,EPSAG/1.D3/
      DATA ERSMAX/1.D15/
C
      IFEP = 1
C
C   INITIALIZE COMMON AREA  /DINCOM/
C
      CALL INIT(N,X0,H,EPS,DX,IRAND,F,
     1    NTRAJ,ISEGBR,INKPBR,KPBRO,INHP,IFEP,XRMIN,XRMAX,IOUT)
C
C   CHECK PARAMETER VALUES
C
      IF(NPRIN.LE.0.OR.NPMAXO.LT.0.OR.INPMAX.LE.0.OR.
     1    NSUC.LE.0.OR.NTRIAL.LE.0)IOUT = -99
      IF (IOUT.EQ.(-99))RETURN
C
C   INITIALIZE VARIABLES
C
      EPSC = EPS
```

```
      447.          NPMAX = NPMAX0
      448.          ISTOPT = 0
      449.          ISTOP = 0
      450.          ICCOM = (NTRIAL+NTRIAL**4)/5
      451.          NFEV = 0
      452.          NTES = NSUC
      453.          ICTS = NSUC-1
      454.          FTFOPT = F
      455.    C
      456.    C  START SERIES OF TRIALS
      457.    C
      458.          DO 30 IC = 1,NTRIAL
      459.    C
      460.    C  SET INITIALIZATION INDEX FOR NOISE GENERATOR
      461.    C
1     462.             IS = 0
1     463.             IF(IRAND.GT.0)IS = IRAND+IC-1
1     464.    C
1     465.    C  INITIALIZE TRIAL
1     466.    C
1     467.             IF(IC.GT.1.AND.IC.LE.ICCOM)CALL REINIT(N,X0,EPSC,IS,F,IFEF)
1     468.             IF(IC.GT.ICCOM)CALL REINIT(N,XMIN,EPSC,IS,FOPT,IFEP)
1     469.    C
1     470.             FTFMIN = F
1     471.             FTFMAX = F
1     472.             NFEV = NFEV+1
1     473.    C
1     474.    C  PRINT INITIAL CONDITIONS OF TRIAL
1     475.    C
1     476.             IF(IPRINT.GT.0)CALL PTSEG(N,X0,FTFMIN,FTFMAX,0,NFEV)
1     477.    C
1     478.    C
1     479.    C  DEACTIVATE SCALING
1     480.    C
1     481.             IF(KPASCA.GT.NPMAX.OR.N.LE.1)CALL NOSCA
1     482.    C
1     483.    C  INITIALIZE COMMON AREA /SCALE/
1     484.    C
1     485.             IF(KPASCA.LE.NPMAX.AND.N.GT.1)CALL INISCA(N,NTRAJ)
1     486.    C
1     487.    C  PERFORM A TRIAL
      488.    C
      489.             CALL TRIAL(N,NPMIN,NPMAX,KPASCA,TOLREL,TOLABS,
      490.           1       IPRINT,XMIN,FTFMIN,
      491.           1       FTFMAX,NFEV,KP,ISTOP)
      492.    C
      493.    C
      494.    C     EVALUATE PAST TRIAL AND PREPARE NEXT TRIAL
      495.    C
      496.    C
1     497.    C  SET TRIAL DURATION
      498.    C
      499.             IF(ISTOP.EQ.0)NPMAX = NPMAX+INPMAX
      500.    C
      501.    C  RESET CURRENT NUMBER OF SUCCESSES REQUIRED BEFORE STOPPING
1     502.    C  COMPUTE INDICATOR OF TRIAL STOPPING CONDITIONS
      503.    C  UPDATE BEST CURRENT VALUES OF TRIAL STOPPING INDICATOR AND
```

```
C  OF FUNCTION  F(X)  AT TRIAL STOP
C
      IF((FTFMIN.GT.FTFOPT.OR.ISTOP.EQ.0).AND.ISTOPT.NE.0)GO TO 10
      IF(ITOLCH(FTFOPT,FTFMIN,TOLREL,TOLABS).EQ.0)NTES = NSUC
C
      FTFOPT = FTFMIN
      ISTOPT = ISTOP
   10 CONTINUE
      ISTOPT = IABS(ISTOPT)
      CALL RCLOPT(N,XMIN,FOPT)
      IF(ITOLCH(FTFOPT,FOPT,TOLREL,TOLABS).EQ.0)ISTOPT = -ISTOP-1
      IF(ITOLCH(FTFMIN,FOPT,TOLREL,TOLABS).EQ.0)ISTOP = -ISTOP
C
C  END-OF-TRIAL PRINT OUT
C
      IF(IPRINT.GE.0)
     1   CALL PTRIAL ( N, XOPT, FOPT,
     2                 FTFMIN, FTFMAX, FTFOPT,
     3                 ISTOP, ISTOPT, NFEV, KP, IPRINT )
C
C  UPDATE INITIAL VALUE OF NOISE COEFFICIENT FOR NEXT TRIAL
C
      IF (ISTOP.EQ.0)EPSC = EPSC/EPSR
      IF(ISTOP.GT.0)EPSC = EPSC*EPSAG
      IF(ISTOP.LT.0.AND.IC.LE.ICCOM)EPSC = EPSC*EPSAR
      IF(ISTOP.LT.0.AND.IC.GT.ICCOM)EPSC = EPSC/EPSR
C
C  UPDATE OPERATING CONDITIONS FOR SELECTING (IN THE NEXT TRIAL)
C  THE STARTING VALUE OF THE NOISE COEFFICIENT OF THE NEW TRAJECTORY
C  AFTER BRANCHING
C
      IF (ISTOP.EQ.0)IFEP = 1
      IF (ISTOP.NE.0)IFEP = 2
C
C  UPDATE, PRINT, AND CHECK TRIAL STOPPING CONDITIONS
C
      IF(ISTOP.GT.0.AND.ISTOPT.GT.0)NTES = NTES-1
      IF(NTES.LE.0.OR.ISTOPT.LE.0.OR.ISTOP.LE.0.OR.NTES.GT.ICTS.OR.
     1     IPRINT.LT.0) GO TO 20
      KSUC = NSUC-ICTS
      CALL PTKSUC(KSUC)
      ICTS = NTES-1
   20 CONTINUE
      IF(NTES.LE.0.AND.ISTOPT.GT.0.AND.ISTOP.GT.0)GO TO 40
C
C  CONSTRAIN NOISE COEFFICIENT WITHIN BOUNDS
C
      EPSC = DMIN1(EPSC,EPSMAX)
      IF(EPSC.LE.0.D0)EPSC = 1.D0
   30 CONTINUE
C  END OF SERIES OF TRIALS
C
   40 CONTINUE
C
C  INDICATOR OF STOPPING CONDITIONS
C
      IOUT = ISTOPT
```

```
      1206.           CALL DERCEN(NDIM,X,FV,DX,W,DFDX)
      1207.     C
      1208.     C TRY AGAIN THE FIRST HALF-STEP
      1209.           DO 40 IC = 1,NDIM
     1*1210.              XP(IC) = X(IC)-H*W(IC)*DFDX*DBLE(FLOAT(NDIM))
     1*1211.        40    CONTINUE
      1212.           F = FUNCTO(NDIM,XP)
      1213.           FVS = FV+DX*DABS(DEDXV-DFDX)
      1214.     C
      1215.     C
      1216.     C UPDATE STEPLENGTH  H  AND ACCEPT OR REJECT THE HALF-STEP
      1217.           CALL NEWH(KTIM,FVS,F,H,IE,IEC)
      1218.     C
      1219.     C UPDATE CUMULATED PAST SCALING DATA
      1220.           IF(IEC.GE.1)CALL CUMSCA(NDIM,W,DFDX)
      1221.           IF(IEC.GE.1)GO TO 10
      1222.           DX = DX*HDX
      1223.     C
      1224.     C FIRST HALF-STEP ACCEPTED
      1225.        50 CONTINUE
      1226.     C
      1227.     C UPDATE CUMULATED PAST SCALING DATA
      1228.           CALL CUMSCA(NDIM,W,DFDX)
      1229.           FS = FV+DX*DABS(DFDX)
      1230.           IF(ITOLCH(FS,FV,TOLRI,TOLABS).EQ.0)DX = DX/RCD
      1231.           IF(ITOLCH(FS,FV,TOLRA,TOLABS).GT.0)DX = DX*RCD
      1232.           EPSR = DSQRT(H)*EPS
      1233.     C
      1234.     C TAKE A SAMPLE INCREMENT OF THE WIENER PROCESS
      1235.           CALL GAUSRV(NDIM,W)
      1236.     C
      1237.     C TRY THE SECOND HALF-STEP
      1238.           DO 60 IC = 1,NDIM
     1*1239.              XP(IC) = XP(IC)+EPSR*W(IC)
     1*1240.        60    CONTINUE
      1241.           F = FUNCTO(NDIM,XP)
      1242.     C
      1243.     C ACCEPT OR REJECT THE COMPLETE STEP
      1244.           IF (F-FV.LE.EPS*EPS*SDF) GO TO 70
      1245.           H = H*HR
      1246.           IE = IE+1
      1247.           IF (H.GT.HMINS) GO TO 20
      1248.     C
      1249.     C STEP ACCEPTED
      1250.        70 CONTINUE
      1251.           DO 80 IC = 1,NDIM
     1*1252.              X(IC) = XP(IC)
     1*1253.        80    CONTINUE
      1254.           DX = DMIN1(DX,DXMAX)
      1255.           DX = DMAX1(DX,DXMIN)
      1256.     C
      1257.           RETURN
      1258.           END
```

```
1149.    C          COMPUTED
1150.    C    - CALLS THE SUBROUTINE  DERFOR (OR DERCEN ) TO COMPUTE THE FORWARD-
1151.    C          (OR CENTRAL-) DIFFERENCES DIRECTIONAL DERIVATIVE  GAM/N
1152.    C    - CALLS THE SUBROUTINE  NEWH  TO ACCEPT OR REJECT THE FIRST HALF-
1153.    C          STEP AND OBTAIN AN UPDATED VALUE FOR  H
1154.    C    - CALLS THE SUBROUTINE  SUMSCA  TO UPDATE THE CUMULATED SCALING DATA
1155.    C    - UPDATES THE SPATIAL DISCRETIZATION INCREMENT  DX  BASED ON THE
1156.    C          RESULTS OF CALLING THE FUNCTION  ITOLCH
1157.    C    - CALLS THE SUBROUTINE  GAUSRV  TO PERFORM THE SECOND HALF-STEP.
1158.    C
1159.          DOUBLE PRECISION X,H,EPS,DX,F
1160.          DOUBLE PRECISION DFDX,DFDXV,DXMAX,DXMIN
1161.          DOUBLE PRECISION EPSR,FS,FV,FVS,HMINS,HR,HS
1162.          DOUBLE PRECISION RCD,RDX,STF,TOLABS,TOLRA,TOLRI
1163.          DOUBLE PRECISION W,XP
1164.          DOUBLE PRECISION FUNCTO
1165.          DIMENSION X(NDIM)
1166.          DIMENSION W(100),XP(100)
1167.          DATA RDX/1.D-4/
1168.          DATA DXMIN/1.D-35/
1169.          DATA DXMAX/1.D3/
1170.          DATA HR/1.D-1/
1171.          DATA HMINS/1.D-30/
1172.          DATA STF /100.D0/
1173.          DATA RCD/2.D0/
1174.          DATA TOLRI/1.D-5/
1175.          DATA TOLRA/1.D-11/
1176.          DATA TOLABS/0.D0/
1177.    C
1178.          IEC = 0
1179.          FV = F
1180.       10 CONTINUE
1181.       20 CONTINUE
1182.    C
1183.    C TAKE A RANDOM DIRECTION FOR THE DIRECTIONAL DERIVATIVE
1184.          CALL UNITRV(NDIM,W)
1185.    C
1186.    C COMPUTE FORWARD-DIFFERENCE DERIVATIVE
1187.          CALL DERFOR(NDIM,X,FV,DX,W,DFDX)
1188.    C
1189.    C TRY THE FIRST HALF-STEP
1190.          DO 30 IC = 1,NDIM
1 1191.          XP(IC) = X(IC)-H*W(IC)*DFDX*DBLE(FLOAT(NDIM))
1 1192.       30 CONTINUE
1193.          HS = H
1194.          F = FUNCTO(NDIM,XP)
1195.          FVS = FV+DX*DABS(DFDX)
1196.    C
1197.    C UPDATE STEPLENGTH  H  AND ACCEPT OR REJECT THE HALF-STEP
1198.          CALL NEWH(KTIM,FVS,F,H,IE,IEC)
1199.          IF(IEC.LE.0)GO TO 50
1200.          IE = IE-1
1201.          IEC = IEC-1
1202.          H = HS
1203.          DFDXV = DFDX
1204.    C
1205.    C COMPUTE CENTRAL-DIFFERENCES DERIVATIVE
```

```
     1      VMCOR(20),VCOR(20),XRMIN(100),XRMAX(100),XOPT(100),FOPT,
     2      IE(20),ISVT(20,19),KGEN,KTIM,NDIM,NTRAJ,NTRAJR,
     3      ISEGBR,INKPBR,XPBRO,NCF,IFEP,INHP
C
      KTIM = KTIM+1
C
C LOOP ON THE SIMULTANEOUS TRAJECTORY SEGMENTS
      DO 30 ID = 1,NTRAJ
C
C INFORM THE SCALING SUBPROGRAMS (VIA THE COMMON AREA /SCALE/)
C THAT SCALING OPERATIONS ARE TO BE PERFORMED ON SEGMENT  ID .
         CALL SEGSCA(ID)
         F = VCOR(ID)
C
C PERFORM A TIME-INTEGRATION STEP ON SEGMENT  ID .
         KA = KTIM
         NX = NDIM
         DO 10 IX = 1,NX
            XID(IX) = X(IX,ID)
   10       CONTINUE
         HID = H(ID)
         EPSID = EPS(ID)
         DXID = DX(ID)
         IEID = IE(ID)
C
         CALL SSTEP(KA,NX,XID,HID,EPSID,DXID,IEID,F)
C
         DO 20 IX = 1,NX
            X(IX,ID) = XID(IX)
   20       CONTINUE
         H(ID) = HID
         EPS(ID) = EPSID
         DX(ID) = DXID
         IE(ID) = IEID
         VCOR(ID) = F
         VMCOR(ID) = DMIN1(VMCOR(ID),F)
         IF(IK.EQ.1)VMCOR(ID) = F
         IF(KTIM.EQ.1)IE(ID) = 0
   30    CONTINUE
C
      RETURN
      END
```

```
      SUBROUTINE SSTEP(KTIM,NDIM,X,H,EPS,DX,IE,F)
C
C  THE BASIC TIME-INTEGRATION STEP FOR A GIVEN TRAJECTORY IS PERFORMED
C  BY THE SUBROUTINE  STEP  WHICH
C   - CALLS THE FUNCTION  FUNCTO  TO COMPUTE THE VALUE OF  F
C   - CALLS THE SUBROUTINE  UNITRV  TO COMPUTE THE RANDOM DIRECTION
C         ALONG WHICH THE DIRECTIONAL DERIVATIVE  GAM/N  IS TO BE
```

```
1     1051.              DO 40 IDD = 1,NTRAJ
2     1052.                 IF(ISVT(ID,IT1).EQ.ISVT(IDD,IT1))NCV = NCV+1
2     1053.      40         CONTINUE
1     1054.      50      CONTINUE
      1055.           DO 80 IT = 2,NTRAJR
1     1056.              IT1 = IT-1
1     1057.              NCN = 0
1     1058.              DO 70 ID = 1,NTRAJ
2     1059.                 DO 60 IDD = 1,NTRAJ
3     1060.                    IF(ISVT(ID,IT).EQ.ISVT(IDD,IT))NCN = NCN+1
3     1061.      60            CONTINUE
2     1062.      70         CONTINUE
1     1063.              IF(NCN.EQ.NCV)GO TO 110
1     1064.              NCV = NCN
1     1065.      80      CONTINUE
      1066.           DO 90 ID = 1,NTRAJ
1     1067.              IF(ISVT(1,1).NE.ISVT(ID,1))GO TO 100
1     1068.      90      CONTINUE
      1069.           IT = 2
      1070.           GO TO 140
      1071.     100   CONTINUE
      1072.     110   CONTINUE
      1073.     120   CONTINUE
      1074.           IT = IT1+1
      1075.           DO 130 ID = 1,NTRAJ
1     1076.              VMVT(ID,IT) = DMIN1(VMVT(ID,IT),VMVT(ID,IT1))
1     1077.     130      CONTINUE
      1078.     140   CONTINUE
      1079.           DO 160 ITC = IT,NTRAJR
1     1080.              ITM = ITC-1
1     1081.              DO 150 ID = 1,NTRAJ
2     1082.                 VMVT(ID,ITM) = VMVT(ID,ITC)
2     1083.                 ISVT(ID,ITM) = ISVT(ID,ITC)
2     1084.     150         CONTINUE
1     1085.     160      CONTINUE
1     1086.     C
      1087.           RETURN
      1088.           END
```

```
      1089.           SUBROUTINE STEP(IK)
      1090.     C
      1091.     C  STEP  PERFORMS A SINGLE TIME-INTEGRATION STEP FOR EACH ONE OF THE
      1092.     C  SIMULTANEOUS TRAJECTORIES BY REPEATEDLY CALLING THE SUBROUTINE  SSTEP
      1093.     C
      1094.           DOUBLE PRECISION F
      1095.           DOUBLE PRECISION X,H,DX,VMVT,EPS,VMCOR,VCOR
      1096.           DOUBLE PRECISION XAMIN,XRMAX,XOPT,FOPT
      1097.           DOUBLE PRECISION XID,HID,EPSID,DXID
      1098.           DIMENSION XID(100)
      1099.           COMMON /DINCOM/ X(100,20),H(20),DX(20),VMVT(20,19),EPS(20),
```

```
      INTEGER FUNCTION IPRECE(ID1,ID2)
C
C  IPRECE  DETERMINES THE PRECEDENCE RELATION BETWEEN TWO TRAJECTORIES
C  BASED ON THEIR CURRENT VALUE OF  EPS
C
      DOUBLE PRECISION X,H,DX,VMVT,EPS,VMCOR,VCOR
      DOUBLE PRECISION XRMIN,XRMAX,XOPT,FOPT
      COMMON /DINCOM/ X(100,20),H(20),DX(20),VMVT(20,19),EPS(20),
     1    VMCOR(20),VCOR(20),XRMIN(100),XRMAX(100),XOPT(100),FOPT,
     2    IE(20),ISVT(20,19),KGEN,KTIM,NDIM,NTRAJ,NTRAJR,
     3    ISEGBR,INKPBR,KPBRO,NCF,IFEP,INHP
      IPRECE = 0
      IF(KGEN.GT.ISEGBR+INKPBR)GO TO 10
      IF(EPS(ID2).LT.EPS(ID1))IPRECE = ID1
      IF(EPS(ID2).GT.EPS(ID1))IPRECE = ID2
      RETURN
C
   10 CONTINUE
      IF(EPS(ID2).LT.EPS(ID1))IPRECE = ID2
      IF(EPS(ID2).GT.EPS(ID1))IPRECE = ID1
C
      RETURN
      END
```

```
      SUBROUTINE COMPAS
C
C  COMPAS  TAKES CARE OF THE STORAGE OF PAST HISTORY DATA, DISCARDING
C  ALL DATA NOT NEEDED BY THE ONLY USER OF SUCH DATA, THE FUNCTION
C  IPREC .
C
      DOUBLE PRECISION X,H,DX,VMVT,EPS,VMCOR,VCOR
      DOUBLE PRECISION XRMIN,XRMAX,XOPT,FOPT
      COMMON /DINCOM/ X(100,20),H(20),DX(20),VMVT(20,19),EPS(20),
     1    VMCOR(20),VCOR(20),XRMIN(100),XRMAX(100),XOPT(100),FOPT,
     2    IE(20),ISVT(20,19),KGEN,KTIM,NDIM,NTRAJ,NTRAJR,
     3    ISEGBR,INKPBR,KPBRO,NCF,IFEP,INHP
C
      IT1 = 1
      DO 30 IT = 2,NTRAJR
         IT1 = IT-1
         DO 10 ID = 1,NTRAJ
            IF(ISVT(ID,IT).NE.ISVT(ID,IT1))GO TO 20
   10       CONTINUE
         GO TO 120
   20    CONTINUE
   30    CONTINUE
      IT1 = 1
      NCV = 0
      DO 50 ID = 1,NTRAJ
```

```
2       961.      40         CONTINUE
2       962.      50         CONTINUE
1       963.      60      CONTINUE
        964.           IF(IK.EQ.2)GO TO 30
        965.      C
        966.      C RETURN THE INDICES OF THE SEGMENTS WHICH IN THE ORDERING
        967.      C OCCUPY THE FIRST, THE LAST, AND A SUITABLE MEDIUM LEVEL POSITION
        968.      C
        969.           IP = IORD(1)
        970.           IU = IORD(NTRAJ)
        971.           IM = IORD(ISEGBR)
        972.      C
        973.           RETURN
        974.           END
```

```
        975.           INTEGER FUNCTION IPREC(ID1,ID2)
        976.      C
        977.      C  IPREC  DETERMINES THE PRECEDENCE RELATION BETWEEN TWO TRAJECTORIES
        978.      C  BASED ON THE PAST HISTORY DATA
        979.      C
        980.           DOUBLE PRECISION VM1,VM2
        981.           DOUBLE PRECISION X,H,DX,VMVT,EPS,VMCOR,VCOR
        982.           DOUBLE PRECISION XRMIN,XRMAX,XOPT,FOPT
        983.           COMMON /DINCOM/ X(100,20),H(20),DX(20),VMVT(20,199),EPS(20),
        984.          1     VMCOR(20),VCOR(20),XRMIN(100),XRMAX(100),XOPT(100),FOPT,
        985.          2     IE(20),ISVT(20,19),KGEN,KTIM,NDIM,NTRAJ,NTRAJR,
        986.          3     ISEGBR,INKPBR,KPBRO,NCF,IFEP,INHP
        987.      C
        988.           VM1 = VMVT(ID1,NTRAJR)
        989.           VM2 = VMVT(ID2,NTRAJR)
        990.           DO 10 III = 1,NTRAJR
1       991.              II = 1+NTRAJR-III
1       992.              IF(ISVT(ID1,II).EQ.ISVT(ID2,II))GO TO 20
1       993.              VM1 = DMIN1(VM1,VMVT(ID1,II))
1       994.              VM2 = DMIN1(VM2,VMVT(ID2,II))
1       995.        10    CONTINUE
        996.        20 CONTINUE
        997.           IPREC = 0
        998.           IF(VM2.LT.VM1)IPREC = ID2
        999.           IF(VM2.GT.VM1)IPREC = ID1
       1000.      C
       1001.           RETURN
       1002.           END
```

```
                END
      SUBROUTINE ORDER(IR,IM,IU)
C
C  ORDER  COMPARES THE TRAJECTORIES FROM THE POINT OF VIEW OF PAST
C  HISTORY (BY CALLING THE FUNCTION  IPREC ) AND FROM THE POINT OF VIEW
C  OF THEIR CURRENT VALUE OF  EPS  (BY CALLING THE FUNCTION  IPRECE)
C  AND PROVIDES THE TRAJECTORY ORDERING NEEDED FOR SELECTING THE TRA-
C  JECTORY WHICH IS TO BE BRANCHED.
C
      DOUBLE PRECISION X,H,DX,VMVZ,EPS,VMCOR,VCOR
      DOUBLE PRECISION XRMIN,XRMAX,XOPT,FOPT
      COMMON /DINCOM/ X(100,20),H(20),DX(20),VMVT(20,19),EPS(20),
     1    VMCOR(20),VCOR(20),XRMIN(100),XRMAX(100),XOPT(100),FOPT,
     2    IE(20),ISVT(20,19),KGEN,KTIM,NDIM,NTRAJ,NTRAJR,
     3    ISEGBR,INKPBR,KPBRO,NCF,IFEP,INHP
      DIMENSION IORD(20)
      DATA KP/0/
      IR = 0
C
C ASSIGN INITIAL ORDERING
C
   10 CONTINUE
      DO 20 I = 1,NTRAJ
         IORD(I) = I
   20    CONTINUE
C
C SORT TRAJECTORIES ...
C
   30 CONTINUE
      IR = IR+1
C
      DO 60 I = 1,NTRAJR
         I1 = I+1
         DO 50 J = I1,NTRAJ
            K1 = IORD(I)
            K2 = IORD(J)
C
C ... ACCORDING TO PAST HISTORY ...
C
            IF(IR.NE.2)KP = IPREC(K1,K2)
            IF(KP.LE.0.0.AND.IR.EQ.1)GO TO 10
C
C ... OR ACCORDING TO NOISE LEVEL
C
            IF(IR.EQ.2)KP = IPRECE(K1,K2)
            IF(KP.EQ.0)GO TO 40
            KM = K1+K2-KP
            IORD(I) = KP
            IORD(J) = KM
```

```
      C
      C UPDATE PAST HISTORY DATA
      C
            DO 10 ID = 1,NTRAJ
     1         VMVT(ID,NTRAJR) = VMCOR(ID)
     1         ISVT(ID,NTRAJR) = ID
     1   10    CONTINUE
     1  C
     1  C OBTAIN TRAJECTORY-SELECTION ORDERING
     1  C
            CALL ORDER(IP,IM,IU)
      C
      C DECIDE WHICH TRAJECTORY IS TO BE BRANCHED
      C
            IF(MOD(KGEN,INKPBR).EQ.KPBRO)IM = IP
      C
      C PERFORM BRANCHING
      C
            DO 20 IC = 1,NDIM
     1         X(IC,IU) = X(IC,IM)
     1   20    CONTINUE
            H(IU) = H(IM)
            IE(IU) = IE(IM)
            DO 30 IT = 1,NTRAJR
     1         ISVT(IU,IT) = ISVT(IM,IT)
     1         VMVT(IU,IT) = VMVT(IM,IT)
     1   30    CONTINUE
            EPS(IU) = EPS(IM)
            DX(IU) = DX(IM)
            VCOR(IU) = VCOR(IM)
            DO 40 ID = 1,NTRAJ
     1         VMCOR(ID) = VCOR(ID)
     1   40    CONTINUE
     1  C
     1  C UPDATE PAST-HISTORY-DATA MATRICES
     1  C
            CALL COMPAS
      C
      C UPDATE SCALING DATA
      C
            CALL MOVSCA(IU,IM)
      C
      C UPDATE NOISE COEFFICIENT VALUES
      C
            IF (EPS(IU).LE.0.0D0) GO TO 50
            IF(IFEP.EQ.2)
           1    EPS(IU) = EPS(IU)*FACG**(CHAOS(0)-EFAC)
            IF(IFEP.EQ.1)
           1    EPS(IU) = FACG**(DMIN1(DLEPMX,
           1    DLOG10(EPS(IU))+(CHAOS(-1)-EFAC)*DLFACL0)
            EPS(IU) = DMIN1(EPS(IU),EPSMAX)
      50    CONTINUE
      C
      C UPDATE MAGNITUDE VOF SPATIAL DISCRETIZATION INCREMENT
      C
            DX(IU) = DX(IU)*DFAC**CHAOS(0)
            RETURN
```

```
      1     INT(SNGL(DLOG(DBLE(FLOAT(KGEN))+.5D0)/DLOG(2.D0)))+1
       IF(INHP.EQ.2)NKGEN = INT(SNGL(DSQRT(DBLE(FLOAT(KGEN))+.5D0)))
       IF(INHP.EQ.3)NKGEN = KGEN
C
C PERFORM THE REQUIRED NUMBER OF INTEGRATION STEPS
C (FOR ALL TRAJECTORIES)
C
       DO 10 IK = 1,NKGEN
C
C PERFORM A SINGLE INTEGRATION STEP
C (FOR ALL TRAJECTORIES)
C
         CALL STEP(IK)
10     CONTINUE
C
C MANAGE THE TRAJECTORY BRANCHING PROCESS
C
       CALL BRASI
       RETURN
       END
```

```
       SUBROUTINE BRASI
C
C  BHASI  PERFORMS THE SELECTION PROCESS FOR THE TRAJECTORIES
C  BRASI  - UPDATES THE DATA ABOUT THE PAST TRAJECTORIES
C         - ASKS FOR THE TRAJECTORY-SELECTION ORDERING BY CALLING THE
C             SUBROUTINE  ORDER
C         - DISCARDS ONE OF THE TRAJECTORIES
C         - PERFORMS BRANCHING ON ONE OF THE REMAINING TRAJECTORIES
C         - MOVES THE DATA OF THE UNPERTURBED CONTINUATION
C             TO THE POSITION OF THE PERTURBED CONTINUATION
C         - CALLS THE SUBROUTINE  COMPAS  TO EXAMINE DATA ABOUT PAST
C             HISTORY OF THE TRAJECTORIES AND DISCARD IRRILEVANT DATA
C
       DOUBLE PRECISION DBAC,DLEPMX,DLFACL
       DOUBLE PRECISION EBAC,EPSMAX,FACG
       DOUBLE PRECISION CHAOS
       DOUBLE PRECISION X,H,DX,VMVT,EPS,VMCOR,VCOR
       DOUBLE PRECISION XRMIN,XRMAX,XOPT,FOPT
       COMMON /DINCOM/ X(100,20),H(20),DX(20),VMVT(20,19),EPS(20),
      1     VMCOR(20),VCOR(20),XRMIN(100),XRMAX(100),XOPT(100),FOPT,
      2     IE(20),ISVT(20,19),KGEN,KTIM,NDIM,NTRAJ,NTRAJR,
      3     ISEGBR,INKPBR,KPBRO,NCF,IFEP,INHP
C
       DATA FACG/10.D0/
       DATA EFAC/.5D0/
       DATA DFAC/1.D3/
       DATA EPSMAX/1.D15/
       DATA DLFACL /0.30102999566398119400/
       DATA DLEPMX /25.D0/
```

```
1       757.    C
        758.          CALL PERIOD
        759.    C
        760.    C EXTRACT AND RESCALE SOME FINAL VALUES
        761.    C
        762.          FM = VCOR(1)
        763.          FMAX = VCOR(1)
        764.          IFM = 1
        765.          DO 30 ID = 2,NTRAJ
1       766.             FMAX = DMAX1(FMAX,VCOR(ID))
1       767.             IF(VCOR(ID).GE.FM)GO TO 20
1       768.             FM = VCOR(ID)
1       769.             IFM = ID
1       770.       20    CONTINUE
1       771.       30    CONTINUE
        772.          DO 40 IC = 1,NDIM
1       773.             XMIN(IC) = X(IC,IFM)
1       774.       40    CONTINUE
        775.          FMIN = FM
        776.          NCEF = NCF
        777.          CALL SEGSCA(IFM)
        778.          CALL VARSCA(NX,XMIN)
        779.    C
        780.          RETURN
        781.          END
```

```
        782.          SUBROUTINE PERIOD
        783.    C
        784.    C  PERIOD  IS CALLED BY SUBROUTINE  GENEVA  TO PERFORM THE GENERATION
        785.    C  OF THE TRAJECTORY SEGMENTS.
        786.    C  PERIOD  - COMPUTES THE DURATION OF THE OBSERVATION PERIOD, I.E. THE
        787.    C               NUMBER  NNP  OF ACCEPTED INTEGRATION STEPS IN A PERIOD
        788.    C          - COMPUTES ALL THE SEGMENT STEPS BY REPEATEDLY CALLING THE
        789.    C               SUBROUTINE  STEP
        790.    C          - PERFORMS THE SEGMENT SELECTION BY CALLING THE SUBROUTINE
        791.    C               BRASI
        792.    C
        793.          DOUBLE PRECISION X,H,DX,VMVT,EPS,VMCOR,VCOR
        794.          DOUBLE PRECISION XRMIN,XRMAX,XOPT,FOPT
        795.          COMMON /DINCOM/ X(100,200,H(20),DX(20),VMVT(20,199,EPS(20),
        796.         1     VMCOR(20),VCOR(20),XRMIN(100),XRMAX(100),XOPT(100),FOPT,
        797.         2     IE(20),ISVT(20,19),KGEN,KTIM,NDIM,NTRAJ,NTRAJR,
        798.         3     ISEGBR,INKPBR,KPBRO,NCF,IFEP,INNP
        799.    C
        800.    C DETERMINE DURATION OF OBSERVATION PERIOD
        801.    C (NUMBER OF TIME INTEGRATION STEPS)
        802.    C
        803.          KGEN = KGEN+1
        804.          NKGEN = 1
        805.          IF(INNP.EQ.1)NKGEN=
```

```
1       708.    C
1       709.    C PRINT RESULTS OF OBSERVATION PERIOD
1       710.    C
1       711.            IF(IPRINT.GT.0)CALL PTSEG(N,XMIN,FMIN,FMAX,IR,NFEV)
1       712.    C
1       713.    C CHECK TRIAL STOPPING CONDITIONS
1       714.    C
1       715.            IF(IR.LT.NPMIN)GO TO 10
1       716.            ISTOP = ITOLCH(FMAX,FMIN,TOLREL,TOLABS)
1       717.            IF(ISTOP.NE.0)GO TO 30
1       718.    C
1       719.      10    CONTINUE
1       720.      20    CONTINUE
        721.      30  CONTINUE
        722.          CALL NOSCA
        723.    C
        724.          RETURN
        725.          END
```

```
        726.          SUBROUTINE GENEVA(NX,XMIN,FMIN,FMAX,NCEF)
        727.    C
        728.    C  GENEVA  PERFORMS THE GENERATION AND THE FINAL PROCESSING AND
        729.    C  EVALUATION OF THE SET OF TRAJECTORY SEGMENTS CORRESPONDING TO
        730.    C  THE CURRENT OBSERVATION PERIOD.
        731.    C  GENEVA  UPDATES THE SCALING ARRAYS  DIST  AND  BIAS  BY CALLING
        732.    C             THE SUBROUTINES  SEGSCA  AND  UPDSCA
        733.    C          GENERATES THE TRAJECTORY SEGMENTS BY CALLING THE SUB-
        734.    C             ROUTINE  PERIOD
        735.    C          DETERMINES SOME END-OF-SEGMENT RESULTS  (FPFMIN, FPFMAX,
        736.    C             XPFMIN)  USING THE RESCALING CAPABILITIES OF THE SUB-
        737.    C             ROUTINES  SEGSCA  AND  VARSCAL
        738.    C
        739.          DOUBLE PRECISION XMIN,FMIN,FMAX
        740.          DOUBLE PRECISION FM
        741.          DOUBLE PRECISION X,H,DX,VMVT,EPS,VMCOR,VCOR
        742.          DOUBLE PRECISION XRMIN,XRMAX,XOPT,FOPT
        743.          COMMON /DINCOM/ X(300,20),H(20),DX(20),VMVT(20,19),EPS(20),
        744.         1     VMCOR(20),VCOR(20),XRMIN(100),XRMAX(100),XOPT(100),FOPT,
        745.         2     IE(20),ISVT(20,19),KGEN,KTIM,NDIM,NTRAJ,NTRAJR,
        746.         3     ISEGOR,INKPBR,KPBRO,NCF,IFEP,INHP
        747.          DIMENSION XMIN(NX)
        748.    C
        749.    C UPDATE SCALING DATA
        750.    C
        751.          DO 10 ID = 1,NTRAJ
1       752.             CALL SEGSCA(ID)
1       753.             CALL UPDSCA(NX,X(1,ID))
1       754.      10     CONTINUE
1       755.    C
1       756.    C GENERATE THE SIMULTANEOUS TRAJECTORY SEGMENTS
```

```
      659.          KGEN = 0
      660.          KTIM = 0
      661.          DO 30 ID = 1,NTRAJ
     1662.             DO 10 IC = 1,NDIM
     2663.                X(IC,ID) = XG(IC)
     2664.       10    CONTINUE
     1665.             IE(ID) = 0
     1666.             DO 20 IT = 1,NTRAJR
     2667.                ISVT(ID,IT) = 1
     2668.                VMVT(ID,IT) = F
     2669.       20    CONTINUE
     1670.             EPS(ID) = EPS0*EPSV**(ISEGBR-ID)
     1671.             VMCOR(ID) = F
     1672.             VCOR(ID) = F
     1673.       30    CONTINUE
      674.          RETURN
      675.          END
```

```
      676.          SUBROUTINE TRIAL(N,NPMIN,NPMAX,KPASCA,
      677.         1     TOLREL,TOLABS,IPRINT,XMIN,FMIN,
      678.         2     FMAX,NFEV,NR,ISTOR)
      679.    C
      680.    C  THE SUBROUTINE  TRIAL  GENERATES A TRIAL, I.E. A SET OF COMPLETE
      681.    C  SIMULTANEOUS TRAJECTORIES BY REPEATEDLY PERFORMING
      682.    C     A CALL TO THE SUBROUTINE  GENEVA  WHICH GENERATES THE SET OF
      683.    C        SIMULTANEOUS TRAJECTORY SEGMENTS CORRESPONDING TO THE CURRENT
      684.    C        OBSERVATION PERIOD, AND PERFORMS THE TRAJECTORY SELECTION
      685.    C     A (POSSIBLE) CALL TO THE SUBROUTINE  PTSEG  WHICH PERFORMS
      686.    C        END-OF-SEGMENT OUTPUT
      687.    C     A CHECK OF THE TRIAL STOPPING CRITERIA (WITH THE AID OF THE
      688.    CO       FUNCTION  ITOLCH
      689.    C     A DECISION ABOUT ACTIVATING OR DEACTIVATING THE SCALING OF
      690.    C        THE VARIABLES (ACTIONS PERFORMED BY CALLING THE SUBROUTINES
      691.    C        ACTSCA  AND  NOSCA).
      692.    C
      693.          DOUBLE PRECISION TOLREL,TOLABS,XMIN,FMIN,FMAX
      694.          DIMENSION XMIN(N)
      695.          DATA IRNF/7/
      696.    C
      697.          DO 20 IR = 1,NPMAX
      698.    C
      699.    C ACTIVATE SCALING
      700.    C
     1701.             IF(IR.GE.KPASCA.AND.IR.GT.N*IRNF)CALL ACTSCA
     1702.             NR = IR
     1703.    C
     1704.    C GENERATE AND EVALUATE THE SIMULTANEOUS TRAJECTORY SEGMENTS
     1705.    C PERIOD
     1706.    C
     1707.             CALL GENEVA(N,XMIN,FMIN,FMAX,NFEV)
```

```
      610.           IF(ISEGBR.EQ.0)ISEGBR = (1+NTRAJ)/2
      611.           ISEGBR = MINO(ISEGBR,NTRAJ)
      612.           ISEGBR = MAXO(ISEGBR,1)
      613.           N2 = ISEGBR
      614.           INKPBR = N3
      615.           IF(INKPBR.EQ.0)INKPBR = INKPBO
      616.           N3 = INKPBR
      617.           KPBRO = N4
      618.           IF(KPBRO.EQ.0)KPBRO = KPBR00
      619.           KPBRO = MOD(KPBRO,INKPBR)
      620.           N4 = KPBRO
      621.           NDIM = NX
      622.           NTRAJR = NTRAJ-1
      623.           NCF = 1
      624.           F = FUNCT(NX,X0)
      625.           CALL STOOPT(NX,X0,F)
      626.           DO 20 ID = 1,NTRAJ
1     627.              H(ID) = H0
1     628.              DX(ID) = DX0
1     629.     20    CONTINUE
1     630.     C
1     631.     C INITIALIZE REMAINING VARIABLES
1     632.     C
      633.           CALL REINIT(NX,X0,EPS0,IRAND,F,IFE)
      634.     C
      635.           RETURN
      636.           END
```

```
      637.           SUBROUTINE REINIT(NX,X0,EPS0,IRAND,F,IFE)
      638.     C
      639.     C  REINIT  PERFORMS THE INITIALIZATION OF ALL TRIALS FOLLOWING THE
      640.     C  FIRST TRIAL, AND PART OF THE INITIALIZATION OF THE FIRST TRIAL.
      641.     C
      642.           DOUBLE PRECISION X0,EPS0,F
      643.           DOUBLE PRECISION EPSV,G
      644.           DOUBLE PRECISION CHAOS
      645.           DOUBLE PRECISION X,H,DX,VMVT,EPS,VMCOR,VCOR
      646.           DOUBLE PRECISION XRMIN,XRMAX,XOPT,FOPT
      647.           COMMON /DINCOM/ X(100,20),H(20),DX(20),VMVT(20,19),EPS(20),
      648.          1     VMCOR(20),VCOR(20),XRMIN(100),XRMAX(100),XOPT(100),FOPT,
      649.          2     IE(20),ISVT(20,19),KGEN,KTIM,NDIM,NTRAJ,NTRAJR,
      650.          3     ISEGBR,INKPBR,KPBRO,NCF,IFEP,INHP
      651.           DIMENSION X0(NX)
      652.           DATA EPSV/1.D0/
      653.     C
      654.     C INITIALIZE RANDOM NOISE GENERATOR
      655.     C
      656.           G = CHAOS(IRAND)
      657.     C
      658.           IFEP = IFE
```

```
      561.           FMIN = FOPT
      562.     C
      563.           RETURN
      564.     C
      565.           END




      566.           SUBROUTINE INIT(NX,X0,H0,EPS0,DX0,IRAND,F,N1,N2,N3,N4
      567.          1     ,INH,IFE,XRI,XRA,IT)
      568.     C
      569.     C  INIT  PERFORMS THE INITIALIZATION OF THE FIRST TRIAL.
      570.     C  THE PART OF THE INITIALIZATION WHICH IS COMMON ALSO TO THE TRIALS
      571.     C  FOLLOWING THE FIRST ONE IS PERFORMED BY CALLING THE SUBROUTINE
      572.     C  REINIT.
      573.     C
      574.           DOUBLE PRECISION X0,H0,EPS0,DX0,F,XRI,XRA
      575.           DOUBLE PRECISION FUNCT
      576.           DOUBLE PRECISION X,H,DX,VMVT,EPS,VMCOR,VCOR
      577.           DOUBLE PRECISION XRMIN,XRMAX,XOPT,FOPT
      578.           COMMON /DINCOM/ X(100,20),H(20),DX(20),VMVT(20,19),EPS(20),
      579.          1     VMCOR(20),VCOR(20),XRMIN(100),XRMAX(100),XOPT(100),FOPT,
      580.          2     IE(20),ISVT(20,19),KGEN,KTIM,NDIM,NTRAJ,NTRAJR,
      581.          3     ISEGBR,INKPBR,KPBRO,NCF,IFEP,INHP
      582.           DIMENSION X0(NX),XRI(NX),XRA(NX)
      583.           DATA NRMAX/100/
      584.           DATA NTRAJM/20/
      585.           DATA NTRAJ0/7/
      586.           DATA INKPB0/10/
      587.           DATA KPBR00/3/
      588.     C
      589.     C CHECK PARAMETER VALUES
      590.     C
      591.           IT = 0
      592.           IF (NX.GT.NPMAX.OR.NX.LT.1.OR.H0.LE.0.D0.OR.EPS0.LE.0.D0
      593.          1     .OR.DX0.LE.0.D0) IT = -99
      594.           IF (IT.EQ.(-99)) RETURN
      595.     C
      596.     C INITIALIZE SOME VARIABLES
      597.     C
      598.           INHP = INH
      599.           DO 10 IX = 1,NX
1     600.              XRMIN(IX) = XRI(IX)
1     601.              XRMAX(IX) = XRA(IX)
1     602.        10    CONTINUE
      603.           CALL NOSCA
      604.           NTRAJ = N1
      605.           IF(NTRAJ.EQ.0)NTRAJ = NTRAJ0
      606.           NTRAJ = MIN0(NTRAJ,NTRAJM)
      607.           NTRAJ = MAX0(NTRAJ,3)
      608.           N1 = NTRAJ
      609.           ISEGBR = N2
```

```
      SUBROUTINE NEWH(K,FV,F,H,IE,IEC)
C
C  NEWH  IS CALLED BY THE SUBROUTINE  SSTEP  TO DECIDE WHETHER TO ACCEPT
C  OR REJECT THE FIRST HALF-STEP, AND TO PROVIDE AN UPDATED VALUE FOR  H
C
      DOUBLE PRECISION FV,F,H
      DOUBLE PRECISION FA,FR,HMAX,HMIN,R
      DIMENSION FR(3),FA(4)
      DATA FR/1.0500,2.00,10.00/
      DATA FA/1.00,1.100,2.00,10.00/
      DATA HMIN/1.D-30/
      DATA HMAX/1.025/
      DATA IECMAX/50/
C
      IF(FV.LT.F)GO TO 20
C
C STEP ACCEPTED, POSSIBLY INCREASE THE STEPLENGTH  H
C
      R = FA(1)
      IF(IE+2.LT.K)R = FA(2)
      IF(IE+3.LT.K)R = FA(3)
      IF(IE.EQ.0.AND.K.GT.1)R = FA(4)
      H = H*R
   10 CONTINUE
      IEC = 0
      GO TO 30
C
   20 CONTINUE
      IE = IE+1
      IEC = IEC+1
      IF(IEC.GT.IECMAX)GO TO 10
C
C STEP REJECTED, DECREASE  H
C
      IC = MIN0(3,IEC)
      R = FR(IC)
      H = H/R
   30 CONTINUE
      H = DMIN1(H,HMAX)
      H = DMAX1(H,HMIN)
C
      RETURN
      END
```

```
      SUBROUTINE DERFOR(NDIM,X,F,DX,W,DFDX)
C
C  DERFOR  COMPUTES THE FORWARD FINITE-DIFFERNCES DIRECTIONAL
C  DERIVATIVES (CALLING  FUNCND )
C
```

```
      1307.           DOUBLE PRECISION X,F,DX,W,DFDX
      1308.           DOUBLE PRECISION DXF,DXFF,DXMAX,FN,S,XX
      1309.           DOUBLE PRECISION FUNCTO
      1310.           DIMENSION X(NDIM),W(NDIM)
      1311.           DIMENSION XX(100)
      1312.           DATA DXFF/1.D6/,DXB/1.D1/
      1313.           DATA DXMAX/1.D6/
      1314.     C
      1315.        10 CONTINUE
      1316.        20 CONTINUE
      1317.           S = 0.D0
      1318.           DO 30 IC = 1,NDIM
     1  1319.              XX(IC) = X(IC)+W(IC)*DX
     1  1320.              S = S+(XX(IC)-X(IC))**2
     1  1321.        30    CONTINUE
      1322.           IF(S.GT.0.D0)GO TO 40
      1323.           DX = DX*DXFF
      1324.           GO TO 20
      1325.        40 CONTINUE
      1326.           FN = FUNCTO(NDIM,XX)
      1327.           DFDX = (FN-F)/DX
      1328.           IF(DX.GT.DXMAX)RETURN
      1329.           IF(DABS(DFDX).GT.1.D0)RETURN
      1330.           IF(DFDX**2.GT.0.D0)GO TO 50
      1331.           DX = DX*DXF
      1332.           GO TO 10
      1333.        50 CONTINUE
      1334.     C
      1335.           RETURN
      1336.           END
```

```
      1337.           SUBROUTINE DERCEN(NDIM,X,F,DX,W,DFDX)
      1338.     C
      1339.     C  DERFOR  COMPUTED THE CENTRAL FINITE-DIFFERNCES DIRECTIONAL
      1340.     C  DERIVATIVES (CALLING  FUNCTO )
      1341.     C
      1342.           DOUBLE PRECISION X,F,DX,W,DFDX
      1343.           DOUBLE PRECISION FN,FR,XX
      1344.           DOUBLE PRECISION FUNCTO
      1345.           DIMENSION X(NDIM),W(NDIM)
      1346.           DIMENSION XX(100)
      1347.     C
      1348.           DO 10 IC = 1,NDIM
     1  1349.              XX(IC) = X(IC)-W(IC)*DX
     1  1350.        10    CONTINUE
      1351.           FR = FUNCTO(NDIM,XX)
      1352.           FN = F+DFDX*DX
      1353.           DFDX = (FN-FR)/(2.D0*DX)
      1354.     C
      1355.           RETURN
```

```
      END


      SUBROUTINE RCLOPT(N,XO,FO)
C
C  RCLOPT  RECALLS THE CURRENT BEST MINIMUM VALUE  FOPT  FOUND SO FAR
C  FROM ALGORITHM START, AND THE POINT  XOPT  (OR POSSIBLY ONE OF THE
C  POINTS) WHERE  FOPT  WAS OBTAINED
C
      DOUBLE PRECISION XO,FO
      DOUBLE PRECISION X,H,DX,VMVT,EPS,VMCOR,VCOR
      DOUBLE PRECISION XRMIN,XRMAX,XOPT,FOPT
      COMMON /DINCOM/ X(100,20),H(20),DX(20),VMVT(20,19),EPS(20),
     1    VMCOR(20),VCOR(20),XRMIN(100),XRMAX(100),XOPT(100),FOPT,
     2    IE(20),ISVT(20,19),KGEN,KTIM,NDIM,NTRAJ,NTRAJR,
     3    ISEGBR,INKPBR,KPBRO,NCF,IFEP,INHP
      DIMENSION XO(N)
C
      DO 10 I = 1,N
         XO(I) = XOPT(I)
   10    CONTINUE
      FO = FOPT
C
      RETURN
      END


      SUBROUTINE STOOPT(N,XO,FO)
C
C  STOOPT  STORES THE CURRENT BEST MINIMUM VALUE  FOPT  FOUND SO FAR
C  FROM ALGORITHM START, AND THE POINT  XOPT  (OR POSSIBLY ONE OF THE
C  POINTS) WHERE  FOPT  WAS OBTAINED
C
      DOUBLE PRECISION XO,FO
      DOUBLE PRECISION X,H,DX,VMVT,EPS,VMCOR,VCOR
      DOUBLE PRECISION XRMIN,XRMAX,XOPT,FOPT
      COMMON /DINCOM/ X(100,20),H(20),DX(20),VMVT(20,19),EPS(20),
     1    VMCOR(20),VCOR(20),XRMIN(100),XRMAX(100),XOPT(100),FOPT,
     2    IE(20),ISVT(20,19),KGEN,KTIM,NDIM,NTRAJ,NTRAJR,
     3    ISEGBR,INKPBR,KPBRO,NCF,IFEP,INHP
      DIMENSION XO(N)
C
      DO 10 I = 1,N
         XOPT(I) = XO(I)
   10    CONTINUE
```

```
1397.            FOPT = F0
1398.      C
1399.            RETURN
1400.            END



1401.            DOUBLE PRECISION FUNCTION FUNCTO(N,XX)
1402.      C
1403.      C  FUNCTO  IS CALLED  WHENEVER THE VALUE OF THE FUNCTION  F  IS REQUIRED
1404.      C  IN THE NUMERICAL INTEGRATION PROCESS.
1405.      C  THE FUNCTION  FUNCTO
1406.      C    - RESCALES THE VARIABLES BY CALLING THE SUBROUTINE  VARSCA
1407.      C    - CALLS THE SUBROUTINE  RANGE  TO TAKE CARE OF THE CASES WHERE THE
1408.      C        CURRENT POINT  X  IS OUTSIDE A GIVEN ADMISSIBLE REGION
1409.      C    - CALLS THE USER-SUPPLIED FUNCTION  FUNCT  TO ACTUALLY COMPUTE THE
1410.      C        VALUE OF  F
1411.      C    - POSSIBLY CALLS THE SUBROUTINE  STOOPT  TO UPDATE THE CURRENT
1412.      C           BEST MINIMUM FUNCTION VALUE  FOPT  AND THE CORRESPONDING
1413.      C           MINIMIZER  XOPT
1414.      C
1415.            DOUBLE PRECISION XX
1416.            DOUBLE PRECISION F,R,XS
1417.            DOUBLE PRECISION FUNCT
1418.            DOUBLE PRECISION X,H,DX',VMVO',EPS,VMCOR,VCOR
1419.            DOUBLE PRECISION XRMIN,XRMAX,XOPT',FOPT
1420.            COMMON /DINCOM/ X(100,20),H(20),DX(20),VMVT(20,19),EPS(20),
1421.           1     VMCOR(20),VCOR(20),XRMIN(100),XRMAX(100),XOPT(100),FOPT,
1422.           2     IE(20),ISVT(20,19),KGEN,KTIM,NDIM,NTRAJ,NTRAJR,
1423.           3     ISEGBR,INKPBR,KPBRO,NCF,IFEP,INHP
1424.            DIMENSION XX(N)
1425.            DIMENSION XS(100)
1426.      C
1427.            DO 10 IX = 1,N
1428.               XS(IX) = XX(IX)
1429.        10     CONTINUE
1430.      C
1431.      C DESCALE  X-VARIABLES
1432.            CALL VARSCA(N,XS)
1433.      C
1434.      C CONSTRAIN THE  X-VARIABLES WITHIN BOUNDS
1435.            CALL RANGE(N,XS,XRMIN,XRMAX,R)
1436.      C
1437.      C COMPUTE THE FUNCTION VALUE...
1438.            F = FUNCT(N,XS)+R
1439.      C
1440.      C ... AND POSSIBLY UPDATE THE BEST CURRENT MINIMUM
1441.            IF(F.LT.FOPT)CALL STOOPT(N,XS,F)
1442.            FUNCTO = F
1443.            NCF = NCF+1
1444.      C
1445.            RETURN
```

```
      END

      SUBROUTINE RANGE (N,XS,XRMIN,XRMAX,R)
C
C  RANGE  IS CALLED BY THE FUNCTION  FUNCTO  TO TAKE CARE OF THE CASES
C  WHERE THE CURRENT POINT  X  IS OUTSIDE A GIVEN ADMISSIBLE REGION
C
      DOUBLE PRECISION XS,XRMIN,XRMAX,R
      DOUBLE PRECISION A,B,C,D,DLRMAX,RMAX,RR,XC
      DIMENSION XS(N),XRMIN(N),XRMAX(N)
      DATA RMAX /1.D35/
      DATA DLRMAX/80.590478254791599DD/
C
      R = 0.DD
      DO 40 I = 1,N
         A = XRMAX(I)
         C = XRMIN(I)
         XC = XS(I)
         IF (XC.LE.A) GO TO 10
         B = A+A-C
         RR = RMAX
         IF (XC.LT.B) RR = DEXP((XC-A)*DLRMAX/(B-A))-1.DD
         R = R+RR
         XS(I) = XRMAX(I)
         GO TO 30
   10    CONTINUE
         IF(XC.GE.C)GO TO 20
         D = C+C-A
         RR = RMAX
         IF (XC.GT.D) RR = DEXP((C-XC)*DLRMAX/(C-D))-1.DD
         R = R+RR
         XS(I) = XRMIN(I)
   20    CONTINUE
   30    CONTINUE
   40    CONTINUE
C
      RETURN
      END

      INTEGER FUNCTION ITOLCH(FMAX,FMIN,TOLREL,TOLABS)
C
C  ITOLCH  DETERMINES  WHETHER TWO QUANTITIES ARE TO BE CONSIDERED NU-
C  MERICALLY EQUAL WITH RESPECT TO AN ABSOLUTE (OR RELATIVE) DIFFERENCE
```

```
1487.    C  CRITERION, WITHIN GIVEN TOLERANCES
1488.    C
1489.          DOUBLE PRECISION FMAX,FMIN,TOLREL,TOLABS
1490.    C
1491.          ISTOP = 0
1492.    C
1493.    C CHECK RELATIVE DIFFERENCE AGAINST  TOLREL
1494.          IF(DABS(FMAX-FMIN).LE.TOLREL*(DABS(FMIN)+DABS(FMAX))/2.D0)
1495.         1    ISTOP=ISTOP+1
1496.    C
1497.    C CHECK ABSOLUTE DIFFERENCE AGAINST  TOLABS
1498.          IF(FMAX-FMIN.LE.TOLABS)ISTOP = ISTOP+2
1499.          ITOLCH = ISTOP
1500.    C
1501.          RETURN
1502.          END


1503.          SUBROUTINE INISCA(N,ND)
1504.    C
1505.    C  INISCA  INITIALIZES THE COMMON AREA  /SCALE/  FOR THE SCALING DATA
1506.    C
1507.          DOUBLE PRECISION DIST,BIAS,GRAGRA,GRA
1508.          COMMON /SCALE/ DIST(10,10,20),BIAS(10,20),GRAGRA(10,10,20),
1509.         1    GRA(10,20),NGRA(20),LSCA,IDSCA,NX,NORD
1510.          DATA NXMSCA/10/
1511.    C
1512.          LSCA = -1
1513.          IF(N.GT.NXMSCA.OR.N.EQ.1)RETURN
1514.          LSCA = 0
1515.          NX = N
1516.          NORD = ND
1517.          IDSCA = 1
1518.          DO 1 ID = 1,NORD
1519.             DO 2 IX = 1,NX
1520.                DO 3 IV = 1,NX
1521.                   DIST(IX,IV,ID) = 0.D0
1522.                   GRAGRA(IX,IV,ID) = 0.D0
1523.         3         CONTINUE
1524.                DIST(IX,IX,ID) = 1.D0
1525.                BIAS(IX,ID) = 0.D0
1526.                GRA(IX,ID) = 0.D0
1527.          2     CONTINUE
1528.             NGRA(ID) = 0
1529.          1  CONTINUE
1530.    C
1531.          RETURN
1532.          END
```

```
      SUBROUTINE NOSCA
C
C  NOSCA  DEACTIVATES THE SCALING
C
      DOUBLE PRECISION DIST,BIAS,GRAGRA,GRA
      COMMON /SCALE/ DIST(10,10,20),BIAS(10,20),GRAGRA(10,10,20),
     1    GRA(10,20),NGRA(20),LSCA,IDSCA,NX,NORD
C
      LSCA = -1
C
      RETURN
      END
```

```
      SUBROUTINE SEGSCA(ID)
C
C  SEGSCA  SELECTS THE TRAJECTORY WHICH MUST BE RESCALED
C
      DOUBLE PRECISION DIST,BIAS,GRAGRA,GRA
      COMMON /SCALE/ DIST(10,10,20),BIAS(10,20),GRAGRA(10,10,20),
     1    GRA(10,20),NGRA(20),LSCA,IDSCA,NX,NORD
C
      IDSCA = ID
C
      RETURN
      END
```

```
      SUBROUTINE VARSCA(B,X)
C
C  VARSCA  COMPUTES THE RESCALED VARIABLE  AX + B
C
      DOUBLE PRECISION X
      DOUBLE PRECISION XB
      DOUBLE PRECISION DIST,BIAS,GRAGRA,GRA
      COMMON /SCALE/ DIST(10,10,20),BIAS(10,20),GRAGRA(10,10,20),
     1    GRA(10,20),NGRA(20),LSCA,IDSCA,NX,NORD
      DIMENSION X(N),XB(10)
C
      IF(LSCA.LE.0)RETURN
      DO 1 I = 1,N
         XB(I) = BIAS(I,IDSCA)
         DO  2 J = 1,N
            XB(I) = XB(I)+DIST(I,J,IDSCA)*X(J)
```

```
2     1573.        2        CONTINUE
1     1574.        1     CONTINUE
      1575.           DO 3 I = 1,N
1     1576.              X(I) = XB(I)
1     1577.        3     CONTINUE
1     1578.     C
      1579.           RETURN
      1580.           END
```

```
      1581.           SUBROUTINE CUMSCA(N,W,DFDX)
      1582.     C
      1583.     C  CUMSCA  STORES CUMULATED STATISTICAL DATA ON THE REL-CONDITIONING OF
      1584.     C  F(AX+B) W.R.T. X
      1585.     C
      1586.           DOUBLE PRECISION W,DFDX
      1587.           DOUBLE PRECISION DFDXMA
      1588.           DOUBLE PRECISION DIST,BIAS,GRAGRA,GRA
      1589.           COMMON /SCALE/ DIST(10,10,20),BIAS(10,20),GRAGRA(10,10,20),
      1590.          1    GRA(10,20),NGRA(20),LSCA,IDSCA,NX,NORD
      1591.           DIMENSION W(N)
      1592.           DATA DFDXMA/1.D8/
      1593.     C
      1594.           IF(LSCA.LE.0)RETURN
      1595.           IF(DABS(DFDX).GT.DFDXMA)RETURN
      1596.           DO 1 I = 1,N
1     1597.              DO 2 J = 1,N
2     1598.                 GRAGRA(I,J,IDSCA) = GRAGRA(I,J,IDSCA)+DFDX*W(I)*DFDX*W(J)
2     1599.        2        CONTINUE
1     1600.              GRA(I,IDSCA) = GRA(I,IDSCA)+DFDX*W(I)
1     1601.        1     CONTINUE
      1602.           NGRA(IDSCA) = NGRA(IDSCA)+1
      1603.     C
      1604.           RETURN
      1605.           END
```

```
      1606.           SUBROUTINE ACTSCA
      1607.     C
      1608.     C  ACTSCA  ACTIVATES THE RESCALING
      1609.     C
      1610.           DOUBLE PRECISION DIST,BIAS,GRAGRA,GRA
      1611.           COMMON /SCALE/ DIST(10,10,20),BIAS(10,20),GRAGRA(10,10,20),
      1612.          1    GRA(10,20),NGRA(20),LSCA,IDSCA,NX,NORD
      1613.     C
```

```
1614.            IF(LSCA.EQ.0)LSCA = 1
1615.      C
1616.            RETURN
1617.            END
```

```
1618.            SUBROUTINE MOVSCA(IU,IM)
1619.      C
1620.      C  MOVSCA  MOVES THE SCALING DATA OF THE UNPERTURBED CONTINUATION TO THE
1621.      C  POSITION OF THE PERTURBED CONTINUATION
1622.      C
1623.            DOUBLE PRECISION DIST,BIAS,GRAGRA,GRA
1624.            COMMON /SCALE/ DIST(10,10,20),BIAS(10,20),GRAGRA(10,10,20),
1625.           1    GRA(10,20),NGRA(20),LSCA,IDSCA,NX,NORD
1626.      C
1627.            IF(LSCA.LT.0)RETURN
1628.            DO 1 I = 1,NX
1629.  1            DO 2 J = 1,NX
1630.  2               DIST(I,J,IU) = DIST(I,J,IM)
1631.  2               GRAGRA(I,J,IU) = GRAGRA(I,J,IM)
1632.  2            CONTINUE
1633.  1            BIAS(I,IU) = BIAS(I,IM)
1634.  1            GRA(I,IU) = GRA(I,IM)
1635.  1         CONTINUE
1636.            NGRA(IU) = NGRA(IM)
1637.      C
1638.            RETURN
1639.            END
```

```
1640.            SUBROUTINE UPDSCA(N,X)
1641.      C
1642.      C  UPDSCA  UPDATES THE SCALING MATRIX  A  AND THE BIAS VECTOR  B BY
1643.      C  CALLING  EIGSCA  AND  VARSCA
1644.      C
1645.            DOUBLE PRECISION X
1646.            DOUBLE PRECISION AGRAL,ALA1,ALPHA,AMCOR,BIAST
1647.            DOUBLE PRECISION CB,COR,DISTT,SN
1648.            DOUBLE PRECISION EIGSCA
1649.            DOUBLE PRECISION DIST,BIAS,GRAGRA,GRA
1650.            COMMON /SCALE/ DIST(10,10,20),BIAS(10,20),GRAGRA(10,10,20),
1651.           1    GRA(10,20),NGRA(20),LSCA,IDSCA,NX,NORD
1652.            DIMENSION X(N)
1653.            DIMENSION DISTT(10,10),BIAST(10),COR(10,10)
1654.            DATA ALPHA /0.300/
```

```
      1655.    C
      1656.          IF(LSCA.LE.0)RETURN
      1657.          ID = IDSCA
      1658.          IF(NGRA(ID).LT.2*NX+NX)GO TO 2
      1659.          AGRAI = 1.D0/DBLE(FLOAT(NGRA(ID)))
      1660.          AMCOR = 0.D0
      1661.          DO 3 I = 1,NX
1     1662.             DO 4 J = 1,NX
2     1663.             COR(I,J) = AGRAI*GRAGRA(I,J,ID)-AGRAI*GRA(I,ID)*AGRAI*GRA(J,ID)
2     1664.                AMCOR = DMAX1(AMCOR,DABS(COR(I,J)))
2     1665.           4     CONTINUE
1     1666.           3  CONTINUE
      1667.          IF(AMCOR.LE.0.D0)GO TO 12
      1668.          DO 1 I = 1,NX
1     1669.             DO 11 J = 1,NX
2     1670.                COR(I,J) = COR(I,J)/AMCOR
2     1671.          11    CONTINUE
1     1672.           1  CONTINUE
      1673.          ALA1 = EIGSCA(COR)
      1674.          CD = ALA1*(1.D0+ALPHA)
      1675.          DO 5 I = 1,NX
1     1676.             COR(I,I) = COR(I,I)-CD
1     1677.             BIAST(I) = X(I)
1     1678.           5  CONTINUE
      1679.          CALL VARSCA(NX,BIAST)
      1680.          SN = 0.D0
      1681.          DO 6 I = 1,NX
1     1682.             DO 7 J = 1,NX
2     1683.                DISTT(I,J) = 0.D0
2     1684.                DO 8 K = 1,NX
3     1685.                   DISTT(I,J) = DISTT(I,J)-DIST(I,K,ID)*COR(K,J)
3     1686.           8       CONTINUE
2     1687.                SN = SN+DISTT(I,J)**2
2     1688.           7     CONTINUE
1     1689.           6  CONTINUE
      1690.          SN = 1.D0/DSQRT(SN/DBLE(FLOAT(NX)))
      1691.          DO 9 I = 1,NX
1     1692.             BIAS(I,ID) = BIAST(I)
1     1693.             DO 10 J = 1,N
2     1694.                DIST(I,J,ID) = SN*DISTT(I,J)
2     1695.                BIAS(I,ID) = BIAS(I,ID)-DIST(I,J,ID)*X(J)
2     1696.                GRAGRA(I,J,ID) = 0.D0
2     1697.          10    CONTINUE
1     1698.             GRA(I,ID) = 0.D0
1     1699.           9  CONTINUE
      1700.          NGRA(ID) = 0
      1701.           2  CONTINUE
      1702.          12  CONTINUE
      1703.    C
      1704.          RETURN
      1705.          END
```

```
      1706.            DOUBLE PRECISION FUNCTION EIGSCA(COR)
      1707.      C
      1708.      C  EIGSCA COMPUTES THE LARGEST EIGENVALUE OF A MATRIX USED FOR RESCA-
      1709.      C  LING, STARTING FROM A RANDOMLY-CHOSEN ESTIMATE (OBTAINED BY CALLING
      1710.      C  THE SUBROUTINE  UNITRV ) OF THE CORRESPONDING EIGENVECTOR
      1711.      C
      1712.            DOUBLE PRECISION COR
      1713.            DOUBLE PRECISION ALA1,ALA11,ALA10
      1714.            DOUBLE PRECISION PREC,SWW,W,WW
      1715.            DOUBLE PRECISION DIST,BIAS,GRAGRA,GRA
      1716.            COMMON /SCALE/ DIST(10,10,20),BIAS(10,20),GRAGRA(10,10,20),
      1717.           1    GRA(10,20),NGRA(20),LSCA,IDSCA,NX,NORD
      1718.            DIMENSION COR(10,10)
      1719.            DIMENSION W(10),WW(10)
      1720.            DATA PREC /1.D-3/
      1721.            DATA NRMIN /10/
      1722.            DATA NRMAX /100/
      1723.      C
      1724.            CALL UNITRV(NX,W)
      1725.            ALA1 = 0.DO
      1726.            DO 1 IR = 1,NRMAX
     1    1727.               ALA10 = ALA1
     1    1728.               SWW = 0.DO
     1    1729.               DO 2 IX = 1,NX
     2    1730.                  WW(IX) = 0.DO
     2    1731.                  DO 3 JX = 1,NX
     3    1732.                     WW(IX) = WW(IX)+COR(IX,JX)*W(JX)
     3    1733.           3         CONTINUE
     2    1734.                  SWW = SWW+WW(IX)**2
     2    1735.           2      CONTINUE
     1    1736.               ALA1 = DSQRT(SWW)
     1    1737.               ALA11 = 1.DO/ALA1
     1    1738.               IF(IR.GE.NRMIN.AND.ALA1*PREC.GT.DABS(ALA1-ALA10))GO TO 4
     1    1739.               DO 5 IX = 1,NX
     2    1740.                  W(IX) = WW(IX)*ALA11
     2    1741.           5      CONTINUE
     1    1742.           1   CONTINUE
      1743.           4   CONTINUE
      1744.               EIGSCA = ALA1
      1745.      C
      1746.            RETURN
      1747.            END
```

```
      1748.            DOUBLE PRECISION FUNCTION CHAOS(INIT)
      1749.      C
      1750.      C  CHAOS  GENERATES A RANDOM SAMPLE FROM ONE OUT OF FOUR POSSIBLE
      1751.      C PROBABILITY DISTRIBUTIONS USING RANDOM NUMBERS UNIFORMLY
      1752.      C DISTRIBUTED IN (0,1) GENERATED BY THE FUNCTION  UNIFRD
      1753.      C
```

```
1754.    C  INIZ  IS AN INPUT PARAMETER AS FOLLOWS
1755.    C
1756.    C       INIZ>0           INITIALIZATION WITH SEED  INIZ.
1757.    C       INIZ=0           STANDARD GAUSSIAN DISTRIBUTION.
1758.    C       INIZ=-1          CAUCHY DISTRIBUTION.
1759.    C       INIZ=-2          UNIFORM DISTRIBUTION IN (-1,+1).
1760.    C       OTHERWISE        UNIFORM DISTRIBUTION IN ( 0,+1).
1761.    C
1762.          DOUBLE PRECISION UNIFRD,PAI,A,B
1763.    C
1764.          DATA PAI/3.14159265358979324D0/
1765.    C
1766.          IF(INIZ.LE.0) GO TO 10
1767.    C
1768.    C INITIALIZATION.
1769.    C
1770.          CHAOS = UNIFRD(INIZ)
1771.          RETURN
1772.    C
1773.       10 CONTINUE
1774.          A = UNIFRD(0)
1775.          IF(INIZ.NE.0) GO TO 20
1776.          B = UNIFRD(0)
1777.    C
1778.    C GAUSSIAN RANDOM NUMBER BY POLAR METHOD
1779.    C
1780.          CHAOS = DSQRT(-2.D0*DLOG(A))*DCOS(PAI*B)
1781.    C
1782.          RETURN
1783.       20 CONTINUE
1784.    C
1785.    C UNIFORM RANDOM NUMBER IN (0,1)
1786.    C
1787.          CHAOS = A
1788.    C
1789.    C CAUCHY RANDOM NUMBER BY INVERSE TRANSFORMATION
1790.    C
1791.          IF(INIZ.EQ.(-1)) CHAOS = DSIN(PAI*A)/DCOS(PAI*A)
1792.    C
1793.    C UNIFORM RANDOM NUMBER IN (-1,+1)
1794.    C
1795.          IF(INIZ.EQ.(-2)) CHAOS = 2.D0*A-1.D0
1796.    C
1797.          RETURN
1798.          END
```

```
1799.          DOUBLE PRECISION FUNCTION UNIFRD(INIZ)
1800.    C
1801.    C UNIFRD GENERATES THE RANDOM NUMBERS UNIFORMLY DISTRIBUTED IN (0,1)
1802.    C EXPLOITING THOSE GENERATED BY  ALKNUT  WITH A FURTHER RANDOMIZATION
```

```
1803.    C IF THE INPUT PARAMETER INIZ  IS NOT 0
1804.    C THE RANDOM NUMBER GENERATOR IS INITIALIZED
1805.    C
1806.          DOUBLE PRECISION A,B,C,X
1807.          DOUBLE PRECISION XO,P,PO,P1,P2,R1,R2
1808.          DOUBLE PRECISION FINV
1809.    C
1810.          DIMENSION X(61)
1811.    C
1812.          DATA NREM/61/,NRIP/100/
1813.          DATA A,B,C/-1.500,5.500,-2.000/
1814.          DATA FINV/3.5D-5/
1815.          DATA IREM/0/
1816.          DATA PO/3.00/,P1,P2/1.00,3.00/,R1,R2/0.2500,0.7500/
1817.    C
1818.          IF(INIZ.NE.0.OR.IREM.EQ.0) GO TO 10
1819.    C
1820.          IO = IREM
1821.          XO = X(IO)
1822.    C
1823.    C NONLINEARIZATION OF XO TO AVOID LONG-DISTANCE LINEAR RELATIONSHIP.
1824.    C
1825.          IF(XO.GE.FINV)XO = DMOD(1.DO/XO,1.DO)
1826.    C
1827.    C UPDATE A COMPONENT OF THE VECTOR X ...
1828.    C
1829.          CALL ALKNUT(NREM,X,IREM)
1830.    C
1831.    C ... AND FURTHER RANDOMIZE
1832.    C
1833.          UNIFRD = DMOD(XO+X(IO),1.DO)
1834.    C
1835.          RETURN
1836.    C
1837.    C INITIALIZATION OF THE RANDOM NUMBER GENERATOR
1838.    C
1839.       10 CONTINUE
1840.    C
1841.          P = PO-1.00/DBLE(FLOAT(IABS(INIZ))+100.0)
1842.          DO 20 K = 1,NREM
1843.    C
1     1844.             P = C+P*(B+P*A)
1     1845.    C
1     1846.    C
1     1847.             X(K) = R1+(R2-R1)*(P-P1)/(P2-P1)
1     1848.       20    CONTINUE
1     1849.    C
      1850.          IREM = 0
      1851.    C
      1852.          DO 30 K = 1,NRIP
      1853.    C
1     1854.             CALL ALKNUT(NREM,X,IREM)
1     1855.    C
1     1856.       30    CONTINUE
1     1857.    C
      1858.          UNIFRD = X(1)
      1859.    C
```

```
      RETURN
      END


      SUBROUTINE ALKNUT(NREM,X,IREM)
C
C UPDATES THE COMPONENT IREM OF THE NREM-VECTOR X WITH A RANDOM NUMBER
C UNIFORMLY DISTRIBUTED IN (0,1) BY MEANS OF THE ALGORITHM
C OF MITCHELL-MOORE, MODIFIED AS SUGGESTED BY BRENT, QUOTED IN
C D.E.KNUTH, THE ART OF COMPUTER PROGRAMMING, SECOND EDITION,
C SECOND VOLUME, SEMINUMERICAL ALGORITHMS, ADDISON-WESLEY
C PUB. CO., READING (1981), PP. 26-28.
C
      DOUBLE PRECISION X
C
      DIMENSION X(NREM)
C
      DATA N1,N2/24,55/
C
      IF(IREM.NE.0) GO TO 10
C
      IREM = NREM
      I1 = NREM-N1
      I2 = NREM-N2
C
   10 CONTINUE
C
      X(IREM) = DMOD(X(I1)+X(I2),1.D0)
C
      IREM = 1+MOD(IREM,NREM)
      I1 = 1+MOD(I1,NREM)
      I2 = 1+MOD(I2,NREM)
C
      RETURN
      END


      SUBROUTINE GAUSRV(N,W)
C
C GENERATES A RANDOM VECTOR SAMPLE FROM AN N-DIMENSIONAL
C NORMAL DISTRIBUTION
C
      DOUBLE PRECISION W,X,Y,R
      DOUBLE PRECISION CHADS
C
```

3.- APPLICAZIONE DELL'ALGORITMO ALL'ANALISI CONFORMAZIONALE

L'algoritmo descritto nel paragrafo 2 è stato concepito un uso del tutto generale.

Per lo studio delle posizioni di equilibrio delle molecole nel caso di geometria di valenza rigida, la funzione da minimizzare è l'energia conformazionale

$$E(\underline{\phi}) = E(\phi_1 \ldots \phi_m)$$

in cui gli argomenti $\phi_1 \ldots \phi_m$ sono gli angoli di torsione liberi.

Il calcolo effettivo di $E(\underline{\phi})$ viene effettuato come segue.

Dati gli angoli $\phi_1 \ldots \phi_m$ si calcola direttamente il contributo torsionale alla energia $E(\underline{\phi})$, le coordinate cartesiane degli atomi e successivamente i potenziali elettrostatici e quelli di Lennard-Jones tra coppie di atomi non legati nella conformazione in oggetto. La energia $E(\underline{\phi})$ è ottenuta come somma delle energie torsionale, elettrostatica e di Lennard-Jones.

Nella Figura 1 è descritto schematicamente il diagramma di flusso relativo agli algoritmi di calcolo.

Come esempio di applicazione del nuovo algoritmo, consideriamo il frammento (di DNA) di desossiribosio-monofosfato mostrato in Figura 2, che nel seguito indicheremo con la sigla SPS (Sugar-Phosphate-Sugar). Per questo frammento Matsuoka, Tosi e Clementi calcolarono, con un metodo quantomeccanico ab-initio[4,5] le energie di cento conformazioni ottenute variando gli angoli di rotazione interna ε, ζ, α, β, γ, e man

Una singola prova è arrestata (al termine di un periodo di osservazione, e dopo aver eliminato la traiettoria peggiore) se tutti i valori finali della f nelle rimanenti traiettorie risultano - entro tolleranze numeriche, ed eventualmente in punti finali diversi - uguali tra loro (arresto "uniforme").

La prova è in ogni caso arrestata, alla fine di un periodo di osservazione, se si è raggiunto un dato numero massimo di periodi di osservazione.

La prova è considerata un successo soltanto nel caso di un arresto uniforme su un valore finale che sia (numericamente) uguale al più basso valore trovato per f dall'inizio dell'algoritmo.

Le prove sono ripetute cambiando i valori di alcuni parametri, e l'intero algoritmo è arrestato, al termine di una prova, se si raggiunge un dato numero di arresti uniformi tutti al livello del migliore valore di f trovato, o in ogni caso se si raggiunge un dato numero massimo di prove.

L'algoritmo considera di aver trovato il minimo globale se si è avuto almeno un arresto uniforme al livello del migliore valore trovato per f.

in cui h_k è la lunghezza del passo di integrazione temporale, $t_k = h_o + h_1 + h_2 + \ldots + h_{k-1}$, $\underline{r}_k$ e $\underline{u}_k$ sono due vettori aleatori in n-dimensioni scelti il primo da una distribuzione uniforme sulla sfera unitaria e il secondo da una distribuzione gaussiana standard, e $\tilde{\eta}_k$ è una approssimazione a differenze finite della derivata direzionale nella direzione $\underline{r}_k$.

L'algoritmo considera un numero fisso di traiettorie generate dalla (2), che si sviluppano (simultaneamente ma indipendentemente una dall'altra), a partire dalle stesse condizioni iniziali, durante un "periodo di osservazione" in cui il coefficiente di rumore di ogni traiettoria è mantenuto costante, mentre h_k e il passo Δx_k usato per calcolare $\tilde{\eta}_k$ sono aggiustati automaticamente per ciascuna traiettoria.

Al termine di ogni periodo di osservazione le traiettorie sono confrontate, una di esse viene scartata, tutte le altre continuano imperturbate nel periodo di osservazione seguente, e una di esse è prescelta per dare luogo a una "diramazione", e cioè a una seconda continuazione della stessa traiettoria, che differisce dalla prima solo per i valori iniziali di ϵ e Δx, ma che si considera avere la stessa "storia passata" della prima.

Il numero totale di traiettorie simultanee rimane perciò invariato, e la seconda continuazione prende - dal punto di vista del programma di calcolo - il posto della traiettoria scartata.

L'insieme delle traiettorie simultanee è considerato come una singola "prova", e l'algoritmo completo è un insieme di prove ripetute.

L'equazione (1) si può considerare come caso limite dell'equazione (del 2° ordine) di Langevin, quando si trascura il termine inerziale.

L'uso dell'equazione (1) è suggerito dal comportamento per t molto grande, del processo aleatorio $\underline{x}(t)$ soluzione dell'equazione (1) con ε costante, a partire da un punto iniziale $\underline{x}_o$: si ha infatti che - sono ipotesi molto poco restrittive sulla funzione f - la densità di probabilità di $\underline{x}(t)$ all'istante t tende , per $t \longrightarrow \infty$, a una densità di probabilità limite

$$p(\underline{x}) = A\, e^{-\frac{2}{\varepsilon^2} f(\underline{x})}$$

indipendentemente dal punto iniziale $\underline{x}_o$ (A è una costante di normalizzazione), che risulta tanto più concentrata intorno ai minimi globali di f quanto minore è ε , fino a diventare, nel limite per $\varepsilon \longrightarrow 0$, una somma pesata di delta di Dirac centrate sui minimi globali (p. es. in una dimensione (N = 1) se f(x) ha due minimi globali a e b la densità p(x) tende, per $\varepsilon \longrightarrow 0$, alla densità

$$\gamma\, \delta(x - a) + (1 - \gamma)\, \delta(x - b)$$

ove $\gamma = (1 + \sqrt{\beta/\alpha})^{-1}$ essendo $\alpha = f''(a) > 0$ e $\beta = f''(b) > 0$).

Dato il punto iniziale $\underline{x}_o$ e la discretizzazione usata per la (1) ha la forma

$$(2)\quad \underline{x}_{k+1} = \underline{x}_k - h_k\, \eta\, \tilde{\eta}_k\, \underline{r}_k + \varepsilon(t_k)\sqrt{h_k}\, \underline{u}_k \qquad k = 0,1,2,\ldots$$

2.- DESCRIZIONE DELL'ALGORITMO PER LA RICERCA DEL MINIMO GLOBALE

L'algoritmo di minimizzazione globale adottato, che si basa sul metodo proposto da F. Aluffi-Pentini, V. Parisi, e F. Zirilli in [1], è descritto in dettaglio in [2], mentre la sua traduzione in un insieme di sottoprogrammi FORTRAN è in [3].

Il metodo ricerca un punto di minimo globale di una funzione $f(\underline{x}) = f(x_1, x_2, \ldots, x_n)$ di n variabili reali, seguendo le traiettorie generate da una opportuna discretizzazione numerica dell'equazione differenziale stocastica.

$$(1) \qquad d\underline{x} = - \nabla f \; dt + \epsilon \; d\underline{w}$$

[a partire da una condizione iniziale $\underline{x}(0) = \underline{x}_0$] essendo ∇f il gradiente di $f(\underline{x})$, $\underline{w}(t)$ un processo stocastico di Wiener standardizzato a n dimensioni, e ϵ un coefficiente positivo ("coefficiente di rumore") che consideriamo variabile nel tempo.

L'equazione (1) - generalmente considerata con ϵ costante - è nota come equazione di Smoluchowski e Kramers, ed è usata p. es. nello studio della diffusione degli atomi nei cristalli, o nello studio di certe reazioni chimiche.

In queste applicazioni l'equazione (1) rappresenta una diffusione attraverso barriere di potenziale sotto l'azione di una forza aleatoria $\epsilon \; d\underline{w}$, essendo f il potenziale e $\epsilon = \frac{2kT}{m}$, ove k è la costante di Boltzmann, T la temperatura assoluta, e m un coefficiente di massa.

potuto colmare tale lacuna grazie all'applicazione di un nuovo metodo, ispirato alla termodinamica statistica, con il quale il minimo globale è ottenuto numericamente seguendo le traiettorie di un sistema di equazioni differenziali stocastiche.

Nel paragrafo 2 viene descritto il metodo di minimizzazione globale adottato, nel paragrafo 3 il metodo viene applicato allo studio dell'energia conformazionale delle molecole ed in particolare si considera un esempio; nel paragrafo 4 vengono tratte alcune semplici conclusioni.

prietà di raggiungere un minimo locale, che spesso è il più vicino al punto da cui si intraprende la minimizzazione, e di non poter fornire quindi alcuna garanzia che l'energia della conformazione finale trovata sia la più bassa possibile per la molecola in questione.

L'unico modo che, in linea di principio, assicura il raggiungimento del cosiddetto minimo globale è l'esplorazione "a tappeto" dell'iperspazio conformazionale: una possibilità puramente teorica, in quanto il numero di valutazioni della funzione $E(\underline{x})$ che dovrebbero essere eseguiti richiederebbe tempi di calcolo proibitivamente elevati.

Questo discorso vale anche se si mantiene rigida la geometria di valenza (lunghezze di legame ed angoli di valenza costanti) e ci si limita all'esplorazione del sottospazio torsionale: si consideri ad esempio che, per una molecola nella quale l'energia $E(\underline{x})$ dipende da cinque angoli di rotazione interna una esplorazione sufficientemente accurata da poter sperare di trovare tutti i possibili minimi richiede che l'energia sia calcolata ad intervalli angolari di al più 15° per ogni legame ruotato: questo comporta il calcolo dell'energia in circa 8×10^6 punti dell'iperspazio conformazionale. Se gli atomi contenuti nella molecola sono qualche decina il numero di valutazioni di funzioni di E, ed i tempi di calcolo, anche su elaboratori (scalari) di notevole potenza, sono proibitivi. La conclusione che ne consegue è che non è possibile trovare in modo diretto il minimo globale delle funzioni di energia potenziale intramolecolare.

Come verrà mostrato in questa comunicazione, abbiamo

1.- INTRODUZIONE

Uno dei problemi più importanti nello studio teorico delle proprietà molecolari è costituito dalla ricerca delle strutture più stabili di una molecola. Tali strutture corrispondono ai punti dell'iperspazio conformazionale, definito dal vettore delle coordinate interne $\underline{x} \equiv (x_1, x_2, \ldots, x_{3N-6})$ (dove N è il numero di atomi contenuti nella molecola), nei quali l'energia potenziale intramolecolare $E(\underline{x})$ è minima. Tutti questi punti sono caratterizzati dall'annullamento delle derivate parziali prime di E, e dal fatto che la matrice delle derivate parziali seconde sia una matrice non negativa; detta matrice determina, attraverso la legge di distribuzione di Boltzmann, la popolazione degli stati conformazionali intorno al minimo.

Tra i vari minimi della funzione $E(\underline{x})$ nell'iperspazio conformazionale è di maggiore interesse in chimica o biologia il minimo o i minimi globali cioè i punti $\underline{x}^*$ tali che per ogni $\underline{x}$ sia:

$$E(\underline{x}^*) \leq E(\underline{x})$$

Data l'elevata complessità del problema, la ricerca dei minimi globali della funzione di energia potenziale non può essere eseguita per via analitica e richiede l'impiego di procedimenti numerici. Esistono numerosi programmi di calcolo per la minimizzazione di funzioni; ma, indipendentemente dalle loro caratteristiche peculiari e dalle condizioni ottimali di applicabilità, essi sono contraddistinti dalla pro-

Ricerca di conformazioni di minima energia potenziale intramolecolare mediante un nuovo metodo di minimizzazione globale

Camillo Tosi, Raffaella Pavani, Roberto Fusco[1]
Istituto Guido Donegani s.p.a., Via G. Fauser, 4, 28100 Novara

Filippo Aluffi-Pentini[2]
Dipartimento di Matematica, Università di Bari, 70125 Bari

Valerio Parisi[2]
Dipartimento di Fisica, II Università di Roma (Tor Vergata), 00173 Roma

Francesco Zirilli[2]
Istituto di Matematica, Università di Salerno, 84100 Salerno

(1) Lavoro realizzato con il contributo del progetto finalizzato del CNR Chimica Fine e Secondaria, contratto n. 83.00437.95.

(2) Lavoro realizzato con il contributo del Governo degli Stati Uniti attraverso l'European Research Office della U.S. Army, contratto n. DAJA-37-81-C-0740.

APPENDIX A7

Ricerca di conformazioni di minima energia potenziale intramolecolare mediante un nuovo metodo di minimizzazione globale

(Search for minimum-intramolecular-potential patterns by means of a new method for global minimization)

by C. Tosi, R. Pavani, R. Fusco, F. Aluffi-Pentini, V. Parisi, F. Zirilli

(to appear (in italian) in Rendiconti dell'Accademia Nazionale dei Lincei).

```
@FTN,S ACM/A.FILESAMPLE,TP$1.FILESAMPLE
FTN 10R1A  02/12/85-22:02(0,)
     1.    C
     2.    C  MAIN PROGRAM (SAMPLE VERSION)
     3.    C  CALLS  SIGMA  VIA THE DRIVER SUBROUTINE  SIGMA1
     4.    C
     5.          DOUBLE PRECISION X0,XMIN,FMIN
     6.    C
     7.          DIMENSION X0(2),XMIN(2)
     8.    C
     9.    C TEST PROBLEM DATA
    10.    C
    11.    C PROBLEM DIMENSION
    12.          N = 2
    13.    C
    14.    C INITIAL POINT
    15.          X0(1) = 0.D0
    16.          X0(2) = 0.D0
    17.    C
    18.    C SET INPUT PARAMETERS
    19.          NSUC = 3
    20.          IPRINT = 0
    21.    C
    22.    C CALL DRIVER SUBROUTINE  SIGMA1
    23.          CALL SIGMA1(N,X0,NSUC,IPRINT,XMIN,FMIN,NFEV,IOUT)
    24.    C
    25.          STOP
    26.          END

    27.          DOUBLE PRECISION FUNCTION FUNCT (N,X)
    28.    C
    29.    C COMPUTES THE VALUE AT  X  OF THE SIX-HUMP CAMEL FUNCTION
    30.    C
    31.          DOUBLE PRECISION X,XX,YY
    32.          DIMENSION X(N)
    33.          XX = X(1)*X(1)
    34.          YY = X(2)*X(2)
    35.          FUNCT = ((XX/3.D0-2.1D0)*XX+4.D0)*XX+X(1)*X(2)
    36.         1     +4.D0*(YY-1.D0)*YY
    37.          RETURN
    38.          END
```

```
      C
            STOP
      C
            END
```

```
            DOUBLE PRECISION FUNCTION FUNCT(N,X)
      C
      C COMPUTES THE FUNCTION VALUES OF TEST-PROBLEM  NPROB
      C BY CALLING THE SUBROUTINE  GLOMTF .
      C
            DOUBLE PRECISION X
            DOUBLE PRECISION F
            DIMENSION X(N)
      C
            COMMON /TUN/ NPROB
      C
            CALL GLOMTF(NPROB,N,X,F)
            FUNCT = F
      C
            RETURN
            END
```

```
@FTN,S ACM/A.FILETEST,TPFS.FILETEST
FTN 10R1A  02/12/85-22:02(0.)
     1.    C
     2.    C (ALGORITHM SIGMA)
     3.    C MAIN PROGRAM (TEST VERSION)
     4.    C (CALL  SIGMA  VIA THE DRIVER  SIGMA1 )
     5.    C
     6.          DOUBLE PRECISION FMIN,X0,XMAXGL,XMIN,XMINGL
     7.    C
     8.    C COMMON AREA TO PASS TEST-PROBLEM NUMBER  NPROB
     9.    C TO THE FUNCTION  FUNCT  WHICH WILL COMPUTE
    10.    C THE FUNCTION VALUES OF TEST-PROBLEM  NPROB
    11.    C BY CALLING THE TEST-PROBLEM COLLECTION SUBROUTINE  GLOMTF
    12.    C
    13.          COMMON /TUN/ NPROB
    14.    C
    15.    C  X0      INITIAL POINT
    16.    C  XMIN    FINAL ESTIMATE OF GLOBAL MINIMUM
    17.    C  XMINGL, XMAXGL  MUST BE DIMENSIONED HERE IN ORDER TO CALL
    18.    C          THE PRE-EXISTING SUBROUTINE  GLOMIP.
    19.          DIMENSION X0(100),XMIN(100),XMINGL(100),XMAXGL(100)
    20.    C
    21.      10  CONTINUE
    22.    C
    23.    C INPUT PROBLEM NUMBER
    24.          WRITE(6,20)
    25.      20  FORMAT(//////41H INPUT PROBLEM NUMBER (1 TO 37, 0 = STOP))
    26.          READ(5,30)NPROB
    27.      30  FORMAT(I2)
    28.          WRITE(6,40)NPROB
    29.      40  FORMAT(///18H PROBLEM NUMBER = ,I2///////)
    30.    C
    31.    C TERMINATE OR CONTINUE
    32.          IF(NPROB.EQ.0)GO TO 50
    33.    C
    34.    C CALL  GLOMIP  TO GET PROBLEM DIMENSION  N  AND INITIAL POINT  X0
    35.    C NOTE THAT  GLOMIP  RETURNS ALSO THE BOUNDARIES  XMINGL ,  XMAXGL
    36.    C OF THE OBSERVATION REGION (NOT NEEDED HERE)
    37.          CALL GLOMIP(NPROB,N,X0,XMINGL,XMAXGL)
    38.    C
    39.    C SET  NSUC  SO AS TO HAVE GOOD CHANCES, WITHOUT PROHIBITIVE
    40.    C COMPUTATIONAL EFFORT
    41.          NSUC = 5
    42.    C
    43.    C SET  IPRINT  SO AS TO HAVE A MODERATE OUTPUT
    44.          IPRINT = 0
    45.    C
    46.    C CALL DRIVER SUBROUTINE  SIGMA1
    47.          CALL SIGMA1(N,X0,NSUC,IPRINT,XMIN,FMIN,NFEV,IOUT)
    48.    C
    49.    C GO TO THE NEXT PROBLEM
    50.          GO TO 10
    51.    C
    52.    C END OF TEST PROBLEMS
    53.      50  CONTINUE
    54.          WRITE(6,60)
    55.      60  FORMAT(/22H END OF TEST PROBLEMS /)
```

```
      1901.            DIMENSION W(N)
      1902.      C
      1903.            NN = (N+1)/2
      1904.      C
      1905.            DO 20 I = 1,NN
     1 1906.               II = 1+N-I
     1 1907.      C
     1 1908.        10    CONTINUE
     1 1909.               X = CHAOS(-2)
     1 1910.               Y = CHAOS(-2)
     1 1911.               R = X*X+Y*Y
     1 1912.      C
     1 1913.               IF(R.GT.1.DD) GO TO 10
     1 1914.      C
     1 1915.               R = DSQRT(-2.DD*DLOG(R)/R)
     1 1916.      C
     1 1917.               W(I) = X*R
     1 1918.               W(II) = Y*R
     1 1919.      C
     1 1920.        20    CONTINUE
     1 1921.      C
      1922.            RETURN
      1923.            END




      1924.            SUBROUTINE UNITRV(N,W)
      1925.      C
      1926.      C GENERATES A RANDOM VECTOR UNIFORMLY DISTRIBUTED
      1927.      C ON THE UNIT SPHERE.
      1928.      C
      1929.            DOUBLE PRECISION W,WW
      1930.      C
      1931.            DIMENSION W(N)
      1932.      C
      1933.            CALL GAUSRV(N,W)
      1934.            WW = 0.DD
      1935.      C
      1936             DO 10 I = 1,N
     1 1937.               WW = WW+W(I)**2
     1 1938.        10    CONTINUE
      1939.            WW = 1.DD/DSQRT(WW)
      1940.      C
      1941.            DO 20 I = 1,N
     1 1942.               W(I) = WW*W(I)
     1 1943.        20    CONTINUE
     1 1944.      C
      1945.            RETURN
      1946.            END

END FTN 4145 IBANK 3605 DBANK 14797 COMMON
```

tenendo la geometria di valenza costante (con le lunghezze di legame e gli angoli di valenza risultanti dalla analisi cristallografica della citosina-3'-fosfato[6]). Mediante un procedimento di best-fit con tali energie fu trovata una funzione di potenziale (abbreviata TCM dalle iniziali degli autori del rif.[5]), costituita da un'espressione di Lennard-Jones per le energie di interazione fra atomi non legati e da un'espressione di Pitzer per le energie torsionali intrinseche:

$$E(\underline{\phi}) = \sum_{i>j} \left(-\frac{A_{ij}}{r_{ij}^{6}} + \frac{B_{ij}}{r_{ij}^{12}}\right) + \sum_{i=1}^{5} \frac{1}{2} K_{\phi i} (1 + \cos 3\phi_i),$$

dove i ϕ_i sono gli angoli di torsione $\gamma, \beta, \alpha, \zeta, \epsilon$ espressi in radianti e dove la somma $\sum_{i>j}$ è estesa alle coppie di atomi non legati e dove si è assunto nullo il contributo elettrostatico.

I valori numerici dei parametri A, B e K_ϕ sono riportati nella Tabella 1 del rif.[7]. Si osservi che gli ossigeni del gruppo PO_4^- hanno il coefficiente attrattivo A sensibilmente più elevato, e il coefficiente repulsivo B sensibilmente più basso, degli altri ossigeni: corrispondentemente tanto l'ascissa quanto l'ordinata del punto di minimo della curva energia-distanza sono nettamente inferiori nel primo caso che nel secondo (2,04 Å contro 2,72 Å e -10,59 $KJmol^{-1}$ contro -0,61 $KJmol^{-1}$). C'è quindi da aspettarsi che il potenziale TCM tenda a favorire strutture stabilizzate da legami di idrogeno intramolecolari. Ed infatti l'applicazione del nuovo metodo porta ad un minimo globale, rappresentato in Figura 2 (con ϵ = 176,2°, ζ = 180,0°, α = 122,1°,

β =-96,7°, γ = 55,6°, E = -94,0 $KJmol^{-1}$), caratterizzato da contatti H(C3')...O6 = 1,75 Å, H(C2')...O3 = 1,82 Å, H(C2') ...O5' = 2,47 Å. Si osservi che un procedimento di ricerca "diretta" dei minimi di più bassa energia [9] aveva portato all'individuazione di due conformazioni di bassa energia, la prima con ε = -75°, ζ = 180°, α = 70°, β = -110°, γ = 55°, E = -91,9 $KJmol^{-1}$ e la seconda con ε = -170°, ζ =180°, α = 115°, β =-100°, γ = 55°, E = -90,4 $KJmol^{-1}$. Il minimo globale ottenuto tramite il metodo qui usato costituisce un guadagno energetico di 3,6 $KJmol^{-1}$.

4.- CONCLUSIONI

L'uso di un nuovo algoritmo di minimizzazione delle funzioni di energia potenziale intramolecolare, la cui concezione si distacca radicalmente da quella dei metodi finora proposti, pone una serie di problemi ai quali solo l'espressione acquisita attraverso l'applicazione ad un elevato numero di casi potrà dare una risposta completa. Ci limitiamo qui ad indicarne i principali.

Data la natura non deterministica del nuovo algoritmo la probabilità di individuare il minimo globale tende ad 1 al tendere all'infinito del numero di valutazioni della funzione, e, di conseguenza, nel tempo di calcolo necessario alla loro esecuzione. Per tenere conto di questa peculiarità del programma, è possibile decidere a priori quante volte si vuole che un lancio finisca nel medesimo minimo prima che questo possa essere considerato come il minimo globale. Quanto più elevato è questo numero, tanto più la probabilità di aver trovato il minimo globale si avvicina alla certezza. Nel nostro esempio esso è stato posto uguale a 5.

Un altro punto di notevole importanza è l'accuratezza ottenibile nella valutazione del punto di minima energia. Proprio perchè il suo scopo essenziale è l'individuazione del punto di minimo globale, il programma non raggiunge il grado di accuratezza raggiunto da altri metodi di minimizzazione, tanto che è opportuno,quando si sia individuato il minimo, applicare un metodo più rapido di minimizzazione locale, ad es. un metodo a convergenza quadratica,quale quello di Newton-Raphson,

- BIBLIOGRAFIA

[1] F. Aluffi-Pentini, V. Parisi, F. Zirilli: "Global optimization and stochastic differential equations" in corso di stampa su Journal of Optimization Theory and Applications.

[2] F. Aluffi-Pentini, V. Parisi, F. Zirilli: "A global optimization algorithm using stochastic differential equations" inviato per la pubblicazione a ACM Transations on Mathematical Software.

[3] F. Aluffi-Pentini, V. Parisi, F. Zirilli: "Algorithm SIGMA.A stochastic-integration global minimization algorithm" inviato per la pubblicazione a ACM Transations on Mathematical Software.

[4] O. Matsuoka, C. Tosi, E. Clementi: Biopolymers 17, 33 (1978).

[5] C. Tosi, E. Clementi, O. Matsuoka: Biopolymers 17, 51 (1978).

[6] M. Sundaralingam e L.H. Jensen: J. Mol. Biol., 13, 914 (1965).

[7] C. Tosi e Kj. Rasmussen: Biopolymers, 20, 1059 (1981).

[8] C. Tosi, E. Clementi, O. Matsuoka: Biopolymers 17, 67 (1978).

[9] C. Tosi e E. Pescatori: Gazz. Chim. Ital., 108, 365 (1978).

[10] P.C. Fantucci e C. Tosi: 5th American Conference on Theoretical Chemistry, Jackson (Wyoming), 15-20 giugno 1984.

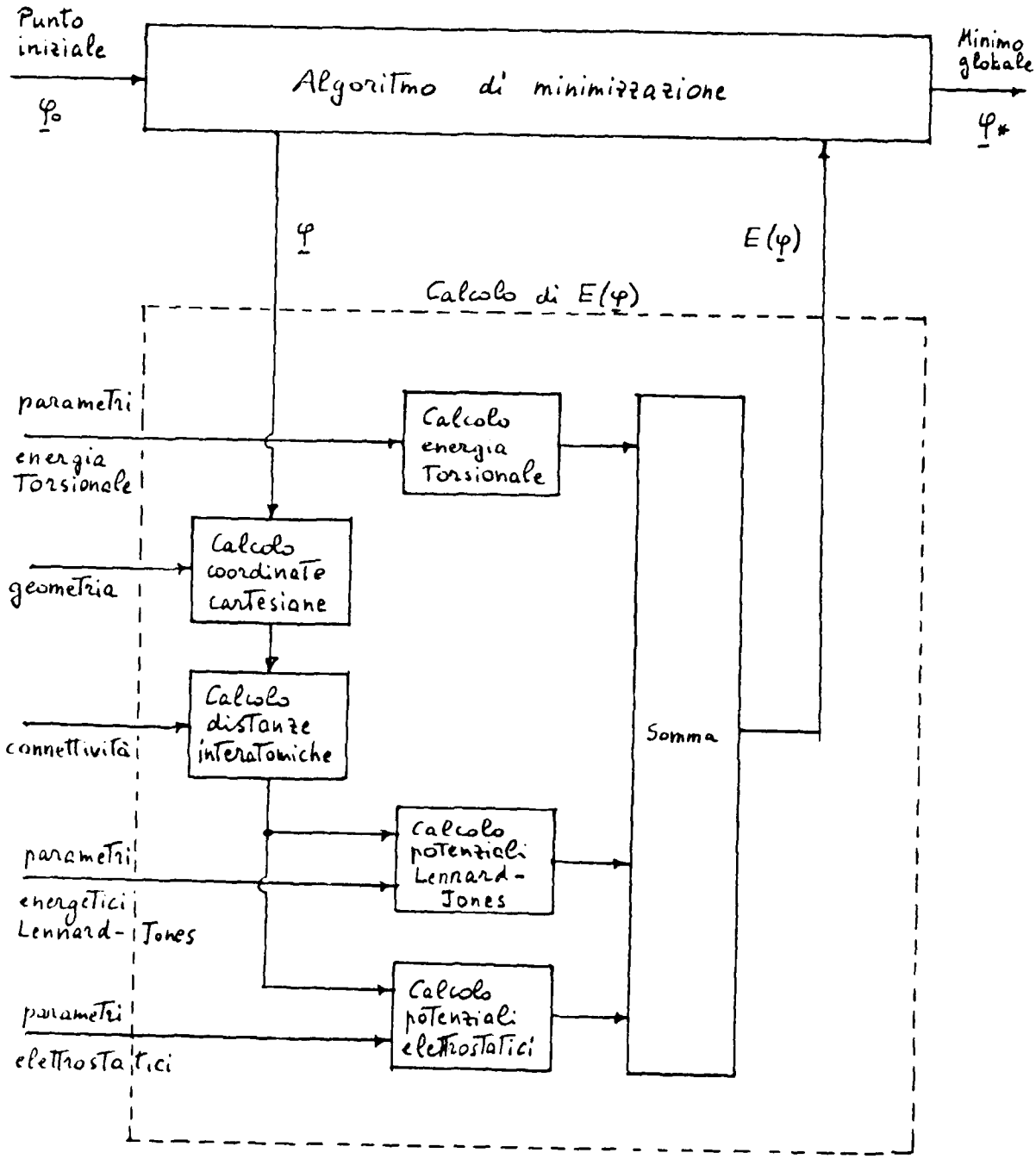


Fig. 1 Diagramma di flusso

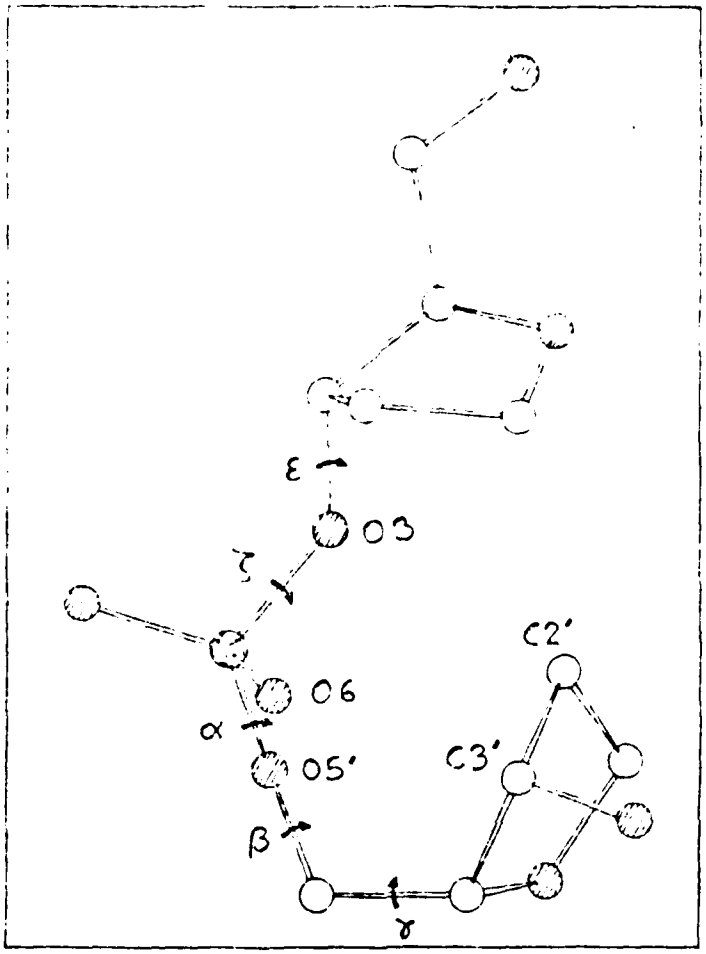


FIGURA 2 - Il frammento C2'(endo) desossi SPS nella conformazione di minima energia trovata con il nuovo algoritmo. I cerchi tratteggiati corrispondono agli atomi di ossigeno e il cerchio pieno all'atomo di fosforo. Per maggiore chiarezza grafica, i 18 atomi di idrogeno sono stati omessi.

END

FILMED

9-85

DTIC