

AD-R157 546

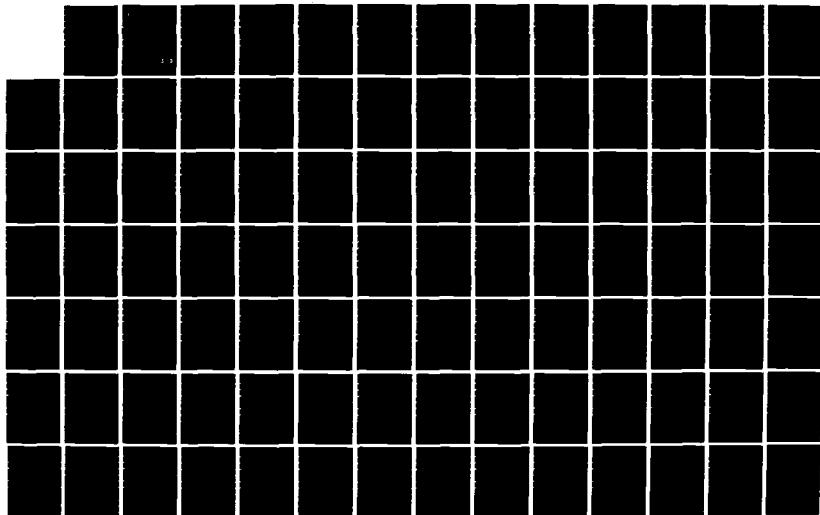
A NEW METHOD FOR GLOBAL OPTIMIZATION BASED ON
STOCHASTIC DIFFERENTIAL EQUATIONS(U) CAMERINO UNIV
(ITALY) MATHEMATICS INST F ALUFFI-PENTINI ET AL.
DEC 84 DAJA37-81-C-0740

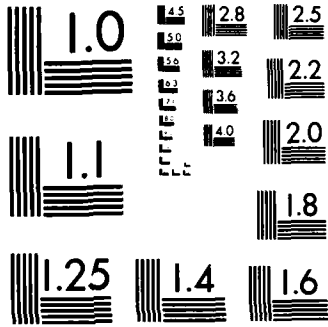
1/3

UNCLASSIFIED

F/G 12/1

NL





MICROCOPY RESOLUTION TEST CHART
NBS-1963-A

AD-A157 546

AD

2

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

DTIC
ELECTE
JUL 23 1985
S D

Approved for Public Release; distribution unlimited

85 07 12 061

DTIC FILE COPY

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
7. AUTHOR(s) Filippo Aluffi-Pentini Valerio Parisi Francesco Zirilli		6. PERFORMING ORG. REPORT NUMBER
9. PERFORMING ORGANIZATION NAME AND ADDRESS Università di Camerino, Istituto Matematico - Via V. Venanzi 2 - Attn. Prof F. Zirilli I-62033 Camerino (MC) Italy		8. CONTRACT OR GRANT NUMBER(s) DAJA37-81-C-0740
11. CONTROLLING OFFICE NAME AND ADDRESS CDR, USARDSG-UK, Box 65, FPO NY 09510		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 6.11.02A 1T161102BH57-05
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		12. REPORT DATE December 1984
		13. NUMBER OF PAGES 300
		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.		

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.

Accession For	
NTIS GRA&I	<input checked="" type="checkbox"/>
DTIC TAB	<input type="checkbox"/>
Unannounced	<input type="checkbox"/>
Justification	
By _____	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
AI	



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

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

1 - Introduction	pag.	1
2 - The proposed method for global optimization	"	2
3 - The algorithm SIGMA	"	5
4 - The software package SIGMA	"	7
5 - Test problems	"	7
6 - Numerical testing	"	8
7 - Application to a problem in theoretical chemistry	"	9
8 - Conclusions	"	9

List of appendices

- 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).
- 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).
- A3 Test problems for global optimization software, by F. Aluffi-Pentini, V. Parisi, F. Zirilli (submitted to ACM Transactions on Mathematical Software).
- A4 A global optimization algorithm using stochastic differential equations, by F. Aluffi-Pentini, V. Parisi, F. Zirilli (submitted to ACM Transactions on Mathematical Software).
- A5 Algorithm SIGMA. A stochastic-integration global minimization algorithm, by F. Aluffi-Pentini, V. Parisi, F. Zirilli (submitted to ACM Transaction on Mathematical Software).
- A6 The FORTRAN package SIGMA.
- 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).

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 optimization

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

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

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)

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

in such a way that

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

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 $\underline{g}: \mathbb{R}^N \rightarrow \mathbb{R}^N$ 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-

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

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

$$(4) \quad \frac{dx}{dt} = - \nabla f(x)$$

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

$$(5) \quad d\xi = -\nabla f(\xi) dt + \epsilon d\omega$$

where ∇f is the gradient of the function f to be minimized, $\omega(t)$ 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 $\epsilon d\omega$, where $\epsilon = (\frac{2kT}{m})^{1/2}$, 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 $\xi(t)$, solution of the equation starting from an initial point x_0 .

It is well known that the probability density function

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

$$(6) \quad p_\epsilon(\underline{x}) = A_\epsilon e^{-(2/\epsilon^2)f(\underline{x})}$$

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 $\epsilon \rightarrow 0$ 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 $t \rightarrow \infty$ of a sample trajectory of a suitable stochastic differential equation it seems natural to try to perform the limit $t \rightarrow \infty$ and the limit $\epsilon \rightarrow 0$ together. We therefore consider the equation (5) with time-varying ϵ , that is

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

with initial condition

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

where

$$(9) \quad \lim_{t \rightarrow \infty} \epsilon(t) = 0.$$

In physical terms condition (9) means that the temperature T is decreased to absolute zero when $t \rightarrow \infty$, 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 $p_\epsilon(\underline{x},t)$ approaches $p_\epsilon(\underline{x})$ for a class of one-dimensional systems, or the rate at which $\epsilon(t)$ 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

$$\underline{\xi}_{k+1} = \underline{\xi}_k - h_k \frac{\partial}{\partial \underline{\xi}} f(\underline{\xi}_k) + \varepsilon(t_k) (h_k)^{\frac{1}{2}} \underline{u}_k, \quad k = 0, 1, 2, \dots$$

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

where h_k is the time integration steplength, $\frac{1}{N} \frac{\partial}{\partial \underline{\xi}} f(\underline{\xi}_k)$ 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 $p_x^{(t)}$ is the probability density of a random variable $\xi_t^{(x)}$ so that $p_x^{(t)} \rightarrow p_x^{(\infty)}$ in law when $t \rightarrow \infty$. Let us remark that $p_x^{(\infty)}$ does not depend on the initial condition x_0 .

We want to study the behaviour of $p_x^{(t)}$ as $\varepsilon_0 \rightarrow 0$ and the rate of approach of $p_x^{(t)}$ to $p_x^{(\infty)}$ as $t \rightarrow \infty$. 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_+, x_+) and increasing in (x_-, x_0) and $(x_+, +\infty)$, with $\lim_{x \rightarrow \pm\infty} f(x) = +\infty$ 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

$$\begin{aligned} f_+ &= f(x_+) & c_+ &= \frac{d^2 f}{dx^2}(x_+) \\ f_- &= f(x_-) & c_- &= \frac{d^2 f}{dx^2}(x_-) \\ f_0 &= f(x_0) & c_0 &= -\frac{d^2 f}{dx^2}(x_0) \\ \Delta f_- &= f_0 - f_- > 0 & \Delta f_+ &= f_0 - f_+ > 0 \end{aligned}$$

it is easy to prove the following result.

Proposition 2.1. Let f be as above and let c_0, c_+, c_- be greater than zero, then

$$\exists \varepsilon_0 > 0, \exists \eta > 0, \exists \lambda > 0 \text{ such that } f(x) \geq \eta(x-x_0)^{-\lambda} + f_0 \quad \forall x \in \mathbb{R}$$

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

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

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

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

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

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

with

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

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

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

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

$$p_{\infty}^{\varepsilon_0}(0, x_0, x) = A_{\varepsilon_0}^{-1} e^{-2f(x)/\varepsilon_0^2} \quad (12)$$

2. Method

Let us consider the Cauchy problem

$$d\xi = -\nabla f(\xi) dt + \varepsilon(t) dw \quad (3)$$

$$\xi(0) = x_0 \quad (4)$$

for the (Ito) stochastic differential equation (3), where $f: \mathbb{R}^n \rightarrow \mathbb{R}$ 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

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

and

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

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

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

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

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 $t \rightarrow \infty$, 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 $t \rightarrow \infty$. 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

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

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 $\xi^{\varepsilon_0}(t)$ is the solution process of (2) starting from an initial point x_0 , then the probability density function of $\xi^{\varepsilon_0}(t)$ approaches, as $t \rightarrow \infty$, the limit density $A_0 e^{-f(x)/\varepsilon_0^2}$ (where A_0 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 $x = (x_1, x_2, \dots, x_n)^T \in \mathbb{R}^n$ and let $f: \mathbb{R}^n \rightarrow \mathbb{R}$ 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:

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

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 $g: \mathbb{R}^n \rightarrow \mathbb{R}^n$ 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, $x = (x_1, x_2, \dots, x_n)^T \in \mathbb{R}^n$ and $f: \mathbb{R}^n \rightarrow \mathbb{R}$ 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.

Global Optimization and Stochastic Differential Equations^{1,2}

F. ALUFFI-PENTINI³, V. PARISI⁴, AND F. ZIRILLI⁵

Communicated by R.A. Tapia

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

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

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

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

⁵Professor, 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

N _{SUC} =	UNIVAC					VAX				
	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 $10^{\pm 30}$ and $10^{\pm 8}$. 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 $|x_i| \leq 10 \quad i = 1, 2, \dots, 10$.

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 $\hat{y}(\xi_k)$ 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 $f_{\text{TFMIN}} > f_{\text{OPT}}$ the trial is again considered unsuccessful; and if $f_{\text{TFMIN}} = f_{\text{OPT}}$ (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.

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

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

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

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

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 $\varepsilon_0 \rightarrow 0$, 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 $t \rightarrow \infty$ 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 $\varepsilon_0 \rightarrow 0$:

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

So that roughly speaking we can imagine:

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

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 $\varepsilon_0 \rightarrow 0$ has been given by Angeletti, Castagnari, Zirilli in Ref. 6.

Remark 2.2. Since $\lambda_1(\varepsilon_0) \rightarrow 0$ as $\varepsilon_0 \rightarrow 0$ 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:

$$\frac{\partial p}{\partial t} = \frac{\partial^2}{\partial x^2} \Delta p + \text{div}(\nabla f p) \quad (17)$$

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

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

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

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

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

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

$$\lim_{t \rightarrow \infty} \varepsilon(t) = 0 \quad (20)$$

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

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

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

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

$$k = 0, 1, \dots$$

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

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

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

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

$$n \langle \nabla f(\xi_k), r \rangle \quad (24)$$

where $n \langle \nabla f(\xi_k), r \rangle$, 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 $\epsilon(t)$ should go to zero very slowly as t goes to infinity so that computing a single path of (3), (4), choosing $\epsilon(t)$ 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 $\epsilon(t)$ 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 $\epsilon(t) = \text{constant}$; $\epsilon(t)$ 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:

$$\begin{array}{ll} x = -3, & f(x) = 7, \\ x = 0, & f(x) = 250, \\ x = 3, & f(x) = 7. \end{array}$$

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 5. 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 $\{(x,y) \mid |x| < 10, |y| < 10\}$.

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 5 but has a unique global minimizer at $(-1.42513, -0.80032)$ where the function f is equal to -186.7309.

Problem 3. 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

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

The function has 6 minima, two of them are global minima and are located at $(-1.818, -1.129)$, $(0.0938, -0.7126)$.

Problems 7-9 are obtained from the following formula:

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

where $x = (x_1, x_2, \dots, x_n)^T$, $y_i = 1 + (x_i - 1)/4$ $i = 1, 2, \dots, n$,

$$k_2 = 10 \quad A_2 = 1.$$

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

$$x_i = 1 \quad i = 1, 2, \dots, n.$$

We penalize the function (25) as follows:

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

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:

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

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

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

$$f(x) = g_3(x) + \sum_{i=1}^n u(x_i, 10, 100, 4) \quad (28)$$

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 13-18 are obtained from the following formula:

$$g_n(x) = k_c (\sin^2 \ell_2 x_1 + \sum_{i=1}^{n-1} (x_i - A_c)^2 (1 + k_c \sin^2 \ell_2 x_{i+1}) + (x_n - A_c)^2 (1 + k_c \sin^2 \ell_1 x_n)) \quad (29)$$

where $k_c = .1$, $k_s = 1$, $A_c = 1$, $\ell_2 = 3$, $\ell_1 = 2$.

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

In the region $\Omega_1 = \{x \in \mathbb{R}^n : -5 \leq x_i \leq 5 \quad i = 1, 2, \dots, n\}$ the function (29) has roughly 15^n local minimizers and a unique global minimizer at $x_i = 1 \quad i = 1, 2, \dots, n$. We penalize the function (29) as follows:

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

or

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

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 $f(x) = \frac{x^4}{4} - \frac{x^2}{2} + 0.1x$, 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 $f(x,y) = \frac{x^4}{4} - \frac{x^2}{2} + 0.1x + \frac{y^2}{2}$, 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 $f(x,y) = 0.5x^2 + .5(1 - \cos 2x) + y^2$, 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 $f(x) = \left\{ \sum_{i=1}^5 ix_i^2 \right\}_i^4$ where $x = (x_1, \dots, x_5)^T$, 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
$$F(x,y) = \prod_{i=1}^{14} \left[\left(\frac{x_i - x}{y} \right) \right]^{1-\delta_i} \left[1 - \left(\frac{x_i - x}{y} \right) \right]^{\delta_i}$$

the data points x_i, δ_i are given by:

x_i	1219	1571	1577	1144	1201	1225	1244
δ_i	0	0	0	1	1	1	1
x_i	1254	1264	1328	1351	1356	1370	1390
δ_i	1	1	1	1	1	1	1

and
$$u(x) = \frac{e^{-x^2/2}}{\sqrt{2\pi}}$$
.

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	185,819	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	156,061	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	54,110	Yes	
24	48,802	Yes	69,512	Yes	

The program is run twice on each problem, the first time with a given stopping criterion. NFI 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_j has been chosen as follows:

$$x_j = 0 \quad \text{for Problems 1-18}$$

$$x_j = 1/50 \quad \text{for Problem 19}$$

$$x_j = (1,0) \quad \text{for Problem 20}$$

$$x_j = (-5,0) \quad \text{for Problem 21}$$

$$x_j = (0,1) \quad \text{for Problem 22}$$

$$x_j = (10^2, 10^3, \dots, 10^{10}) \quad \text{for Problem 23}$$

$$x_j = (-1250, -1000) \quad \text{for Problem 24.}$$

For Problems 19-22 and 24 the initial point x_j 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.

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

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

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

Table 1

(where \cdot' means differentiation)

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

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

$$\begin{aligned}
(2.20) \quad U_\varepsilon(y) &= \frac{1}{2} \left(\frac{1}{2} \left(\frac{df_{2\varepsilon}}{dy} \right)^2 - \frac{d^2 f_{2\varepsilon}}{dy^2} \right) = \\
&= \frac{1}{4} y^2 \left(2a\varepsilon^2 y^2 + \frac{3b\varepsilon}{\sqrt{2}} y + 2c \right)^2 - \frac{1}{2} \left(6a\varepsilon^2 y^2 + \frac{6b\varepsilon}{\sqrt{2}} y + 2c \right) = \\
&= a^2 \varepsilon^4 y^6 + \frac{3ab}{\sqrt{2}} \varepsilon^3 y^5 + \left(\frac{9}{8} b^2 + 2ac \right) \varepsilon^2 y^4 + \frac{3bc}{\sqrt{2}} \varepsilon y^3 + \\
&\quad + (c^2 - 3a\varepsilon^2) y^2 - \frac{3b\varepsilon}{\sqrt{2}} y - c
\end{aligned}$$

In order to understand intuitively the behavior as $\varepsilon \rightarrow 0$ 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 $\varepsilon \rightarrow 0$.

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

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

has five real roots

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

that is, V_ε has a local minimizer at $y = 0$, two global minimizers

$$\text{at } y = \pm \frac{1}{\varepsilon} \left(\frac{4c^2 + \sqrt{4c^4 + 9\varepsilon^2}}{3} \right)^{1/2} \quad \text{and two local maximizers at}$$

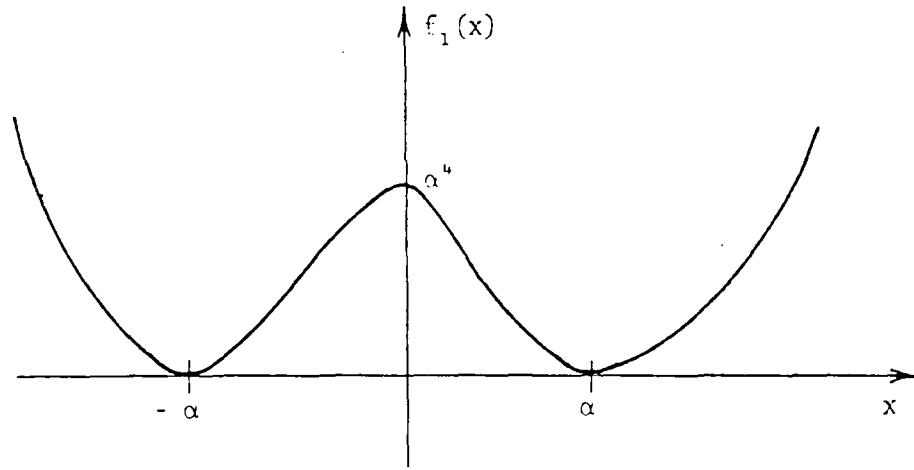


Fig. 1

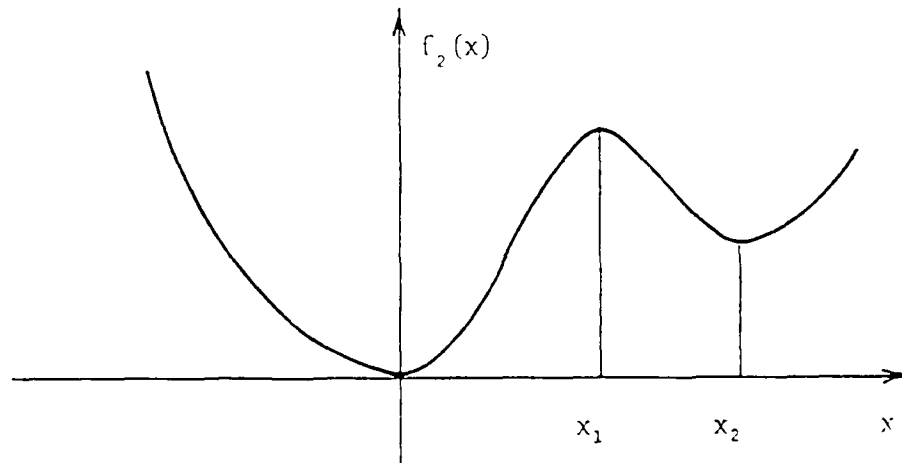


Fig. 2

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

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

or by

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

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

$$(2.15) \quad a > 0$$

$$(2.16) \quad b^2 - 4ac < 0$$

$$(2.17) \quad 9b^2 - 32ac > 0$$

$$(2.18) \quad b < 0$$

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:

$$(2.19) \quad V_\varepsilon(y) = \left. \left\{ \frac{df_2}{dy} \right\}^2 - \frac{d^2f_2}{dy^2} \right\} = \\ = \varepsilon^4 y^6 - 4\alpha^2 \varepsilon^2 y^4 + (4\alpha^4 - 3b^2) y^2 + 2c^2$$

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

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

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

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

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

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

and we will choose

$$(2.11) \quad c_\epsilon = \left[\int_{-\infty}^{+\infty} e^{-2/\epsilon^2 f(x)} dx \right]^{-1}$$

so that

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

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

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

where L_ε is given by (1.4).

Let us consider the change of variables

$$(2.2) \quad y = \frac{\sqrt{2}}{\varepsilon} x$$

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

where c_ε is a normalization constant and

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

The eigenvalue problem (2.1) becomes

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

where

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

and

$$(2.7) \quad W_\varepsilon(y) = \varepsilon^2 \left[\frac{H}{2} - \frac{d^2 F}{dy^2} \right]$$

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 $\varepsilon \rightarrow 0$ of the difference between the resolvent of H_ε and the resolvent of the approximating hamiltonian is proved.

Moreover the asymptotic behavior as $\varepsilon \rightarrow 0$ 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 $P_T\{\cdot\}$ = Probability of $\{\cdot\}$, $p_\varepsilon(x, x_0, t)$ is the solution of the Fokker-Planck equation:

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

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

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

subject to the condition

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

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

The problem of deriving asymptotic formulas as $\varepsilon \rightarrow 0$ for the 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 $\varepsilon \rightarrow 0$ 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 problem 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]:

$$(1.1) \quad dx(t) = -\nabla f(x(t))dt + \varepsilon dw(t)$$

where $f: \mathbb{R} \rightarrow \mathbb{R}$ 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:

$$(1.2) \quad p_\varepsilon(x, x_0, t)dx \equiv P_\varepsilon\{x_\varepsilon(t) \in (x, x+dx) | x_\varepsilon(0) = x_0\}$$

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

Angelo Angeletti

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

Cinzia Castagnari

Istituto Matematico, Università 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 sponsorship 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, Università 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.

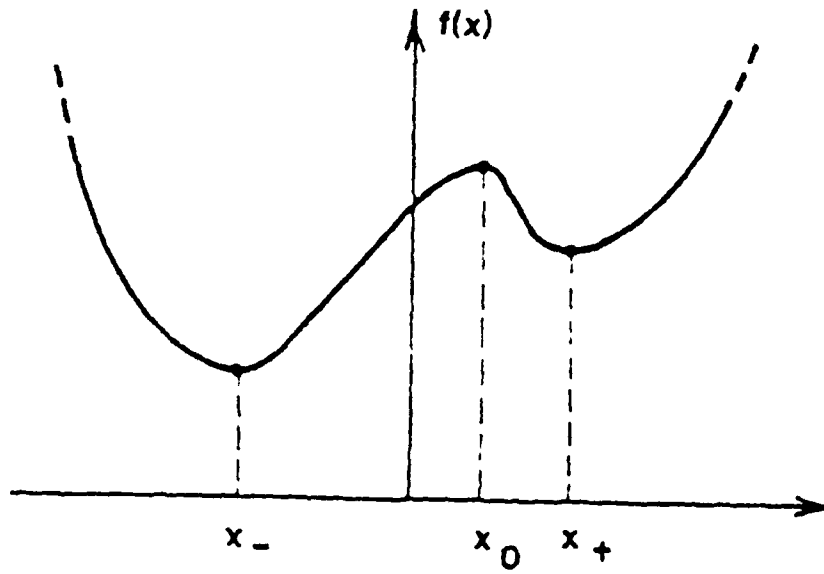


FIGURE 1

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.

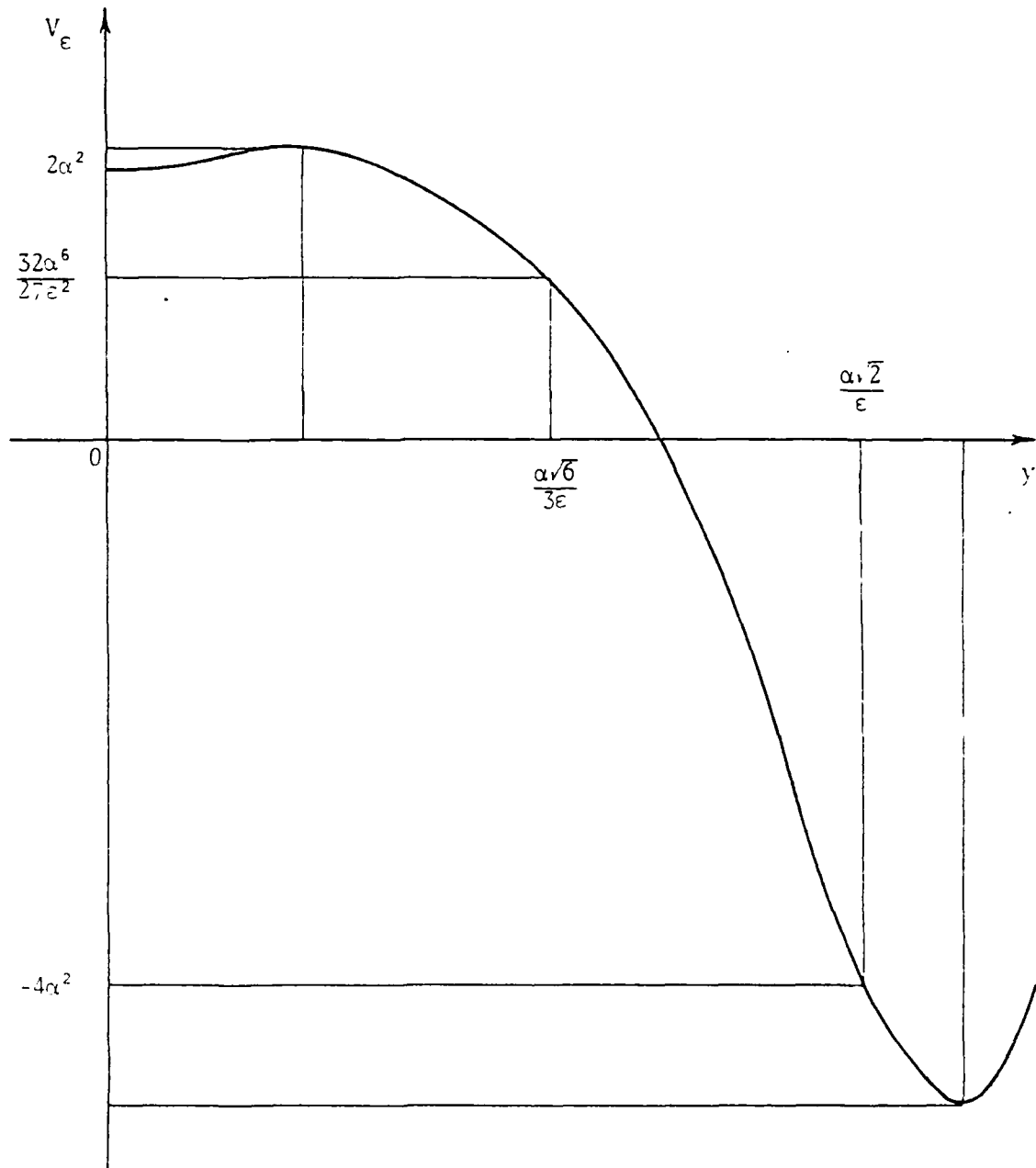


Fig. 3

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 $x_{1,2} = \frac{-3b \mp \sqrt{9b^2 - 52ac}}{8a}$ 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 $\frac{d^2f_2}{dx^2}(\xi_{1,2}) = 0$.

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

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

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

where \cdot' means differentiation.

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

y	0	η_1	x_1	η_2	y_2
$U_2(y)$	$-c$	$\frac{1}{2c^2} (f_1'(\xi_1))^2$	$-c_1$	$\frac{1}{2c^2} (f_2'(\xi_2))^2$	$-c_2$
$U_2'(y)$	$-\frac{3b}{\sqrt{2}}$	$-\frac{1}{\sqrt{2}} f_1'''(\xi_1)$	$-\frac{1}{\sqrt{2}} f_1'''(x_1)$	$-\frac{1}{\sqrt{2}} f_2'''(\xi_2)$	$-\frac{1}{\sqrt{2}} f_2'''(x_2)$
$U_2''(y)$	$2(c^2 - 3ac^2)$	$\frac{1}{2} [f_1''(\xi_1) f_1'''(\xi_1) - \frac{c^2}{2} f_1^{(iv)}(\xi_1)]$	$2c_1^2 - \frac{c^2}{4} f_1^{(iv)}(x_1)$	$\frac{1}{2} [f_2''(\xi_2) f_2'''(\xi_2) - 2c_2^2 - \frac{c^2}{4} f_2^{(iv)}(\xi_2)]$	$2c_2^2 - \frac{c^2}{4} f_2^{(iv)}(x_2)$

Table 2

where $c_1 = \frac{1}{2} \frac{d^2 f_1}{dx^2}(x_1) < 0$, $c_2 = \frac{1}{2} \frac{d^2 f_2}{dx^2}(x_2) > 0$. Moreover $c = 2ax_1 x_2$, $c_1 = 2ax_1(x_1 - x_2)$ and $c_2 = 2ax_2(x_1 - x_2)$.

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

$$\frac{dU_2}{dy} = 0$$

has five real roots so that $U_2(y)$ has three minimizers and two maximizers.

(v) $U_2(y)$ is bounded below by a constant independent of ϵ .

(vi) $U_2(y)$ is given by Fig. 4.

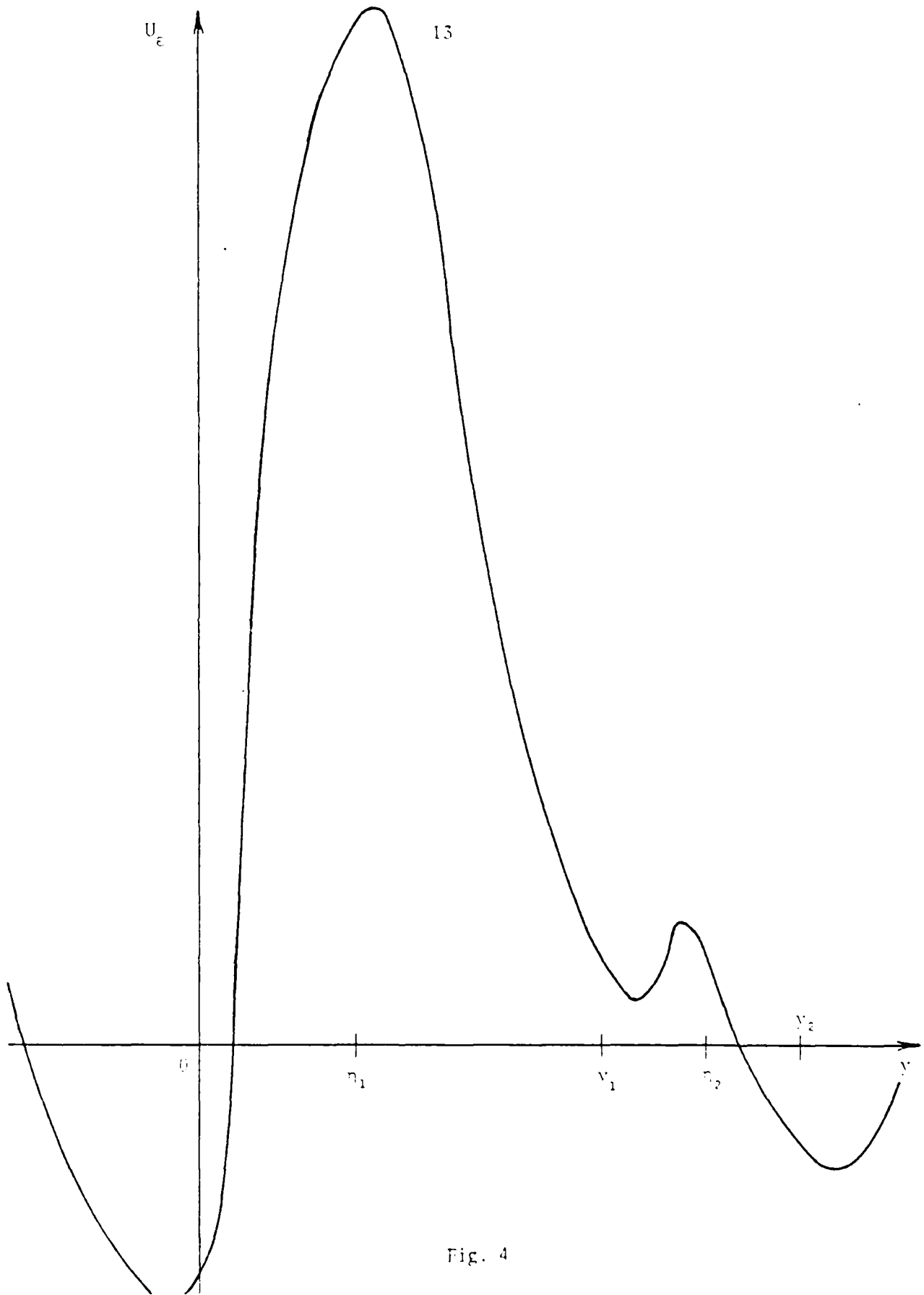


Fig. 4

From Proposition 2.1 and Fig. 3 it follows that as $\varepsilon \rightarrow 0$ $V_\varepsilon(y)$ 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

$$(2.24) \quad -\lambda_n^{(1)} = 4\alpha^2(n+1) \quad n = 0, 1, 2, \dots$$

the eigenvalues corresponding to the remaining two harmonic oscillators are

$$(2.25) \quad -\lambda_n^{(2)} = 8\alpha^2 n \quad n = 0, 1, 2, \dots$$

$$(2.26) \quad -\lambda_n^{(3)} = 8\alpha^2 n \quad n = 0, 1, 2, \dots$$

In section 5 we will prove that the eigenvalues of

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

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

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

as can be seen from (2.24) and

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

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

as can be seen from (2.24), (2.25), (2.26). So that M_ε as $\varepsilon \rightarrow 0$ 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 $\varepsilon \rightarrow 0$ $U_\varepsilon(y)$ 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

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

the eigenvalues corresponding to the remaining two harmonic oscillators are

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

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

In section 5 we will prove that the eigenvalues of

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

approaches (2.30), (2.31), (2.32) when $\varepsilon \rightarrow 0$. In particular we will show that the first eigenvalue $\lambda_1 = 0$ as $\varepsilon \rightarrow 0$ has asymptotically multiplicity 2 (i.e. $\lambda_1(\varepsilon) \rightarrow 0$ when $\varepsilon \rightarrow 0$). 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 $h_0: \mathcal{D}(h_0) \subset L^2(\mathbb{R}) \rightarrow L^2(\mathbb{R})$ 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 $M_\varepsilon, N_\varepsilon$ 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 $A_+ = \{y | y > \frac{\alpha\sqrt{6}}{3\epsilon}\}$, $A_0 = \{y | |y| < \frac{\alpha\sqrt{6}}{3\epsilon}\}$, $A_- = \{y | y < -\frac{\alpha\sqrt{6}}{3\epsilon}\}$
and define $V_{2\epsilon}$ as follows:

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

(see Fig. 5)

where

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

$$(3.3) \quad \beta = \frac{\pi}{2} \frac{3\epsilon}{\alpha\sqrt{6}}$$

$$(3.4) \quad V_0 = \frac{32}{3} \frac{x^6}{\pi^2 \epsilon^2}$$

The function $V_{2\epsilon}$ as $\epsilon \rightarrow 0$ is an approximation to V_ϵ in particular $V_{2\epsilon}$ 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 $y = \pm \frac{\alpha\sqrt{2}}{\epsilon}$ and equations $16\alpha^4 \left(y \mp \frac{\alpha\sqrt{2}}{\epsilon} \right)^2 - 4\alpha^2$.

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

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

$$(3.6) \quad \lim_{\varepsilon \rightarrow 0} \varepsilon \bar{\eta}_1(\varepsilon) = \lim_{\varepsilon \rightarrow 0} \varepsilon \bar{\eta}_2(\varepsilon) = 0$$

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

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

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 $\varepsilon \rightarrow 0$.

Let

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

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

we define

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

(see Fig. 6)

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

Let us now consider the operators

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

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

we will use them to approximate M_{ϵ} .

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

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

can be reduced to the following eigenvalues problems:

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

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

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

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].

$$(3.17) \quad \lambda_{n\epsilon}^{\pm} = 4\alpha^2 \left[2n + \gamma - \frac{\alpha^2}{\nu} \right] \quad n = 0, 1, 2, \dots$$

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

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

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

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

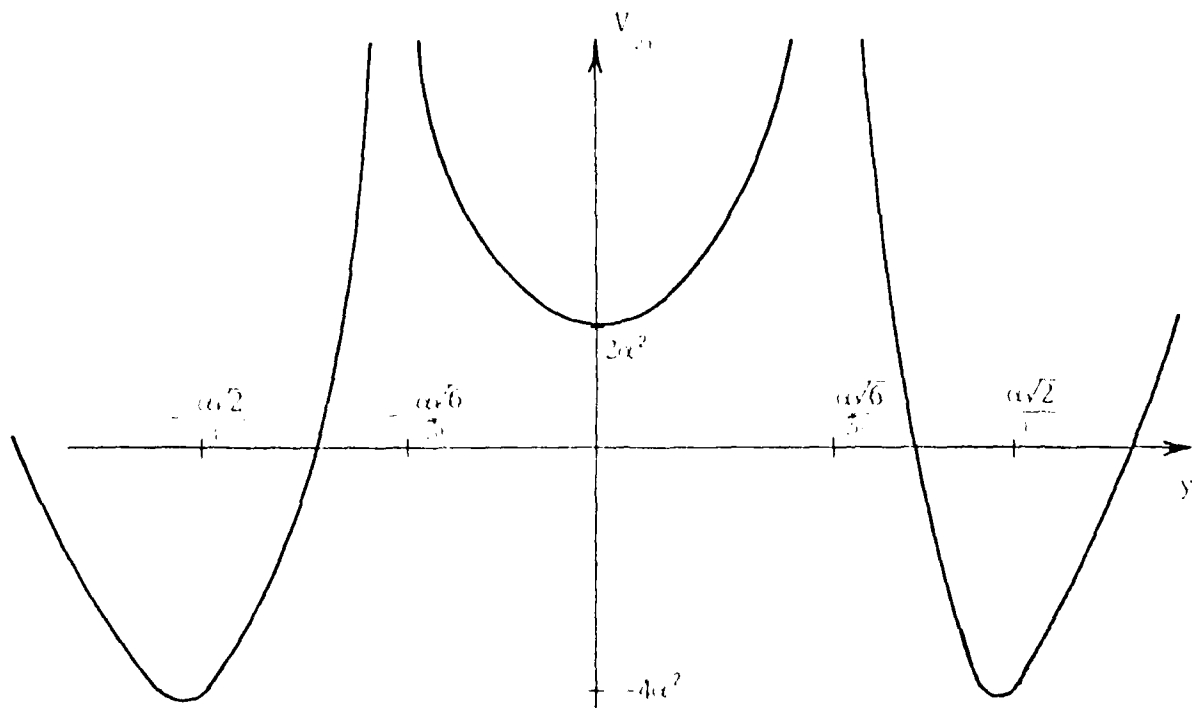


Fig. 5

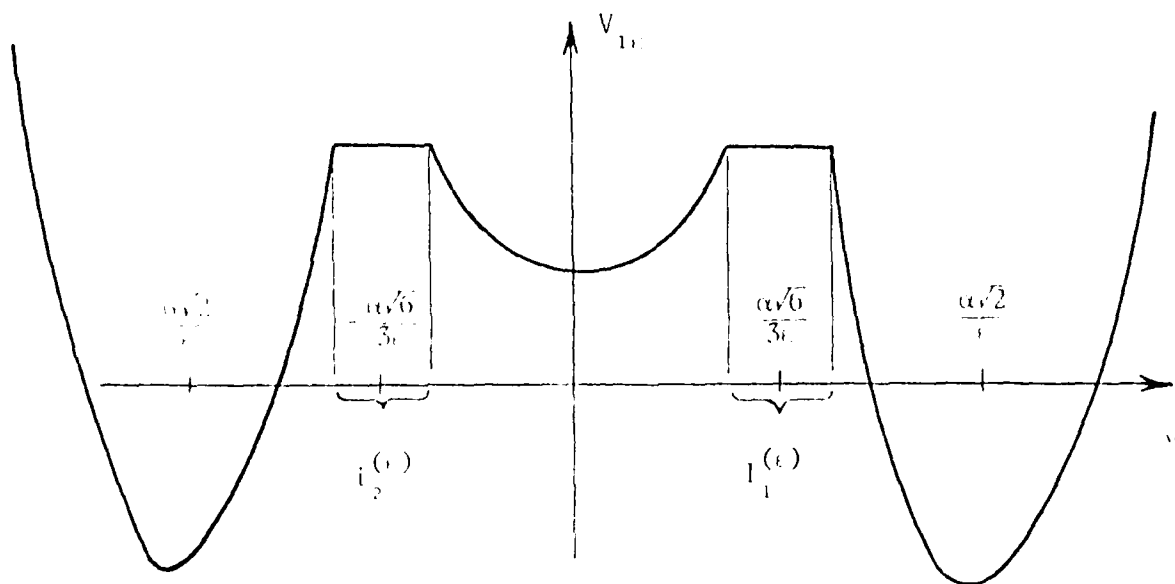


Fig. 6

$$(3.20) \quad \lambda_{n\epsilon}^0 = \beta^2 [n^2 + \delta(2n+1)] + 2\alpha^2 \quad n = 0, 1, \dots$$

$$(3.21) \quad \hat{\lambda}_{n\epsilon}^{\pm} = \begin{cases} \cos^{\delta} \epsilon y F\left(\delta + \frac{n}{2}, -\frac{n}{2}, \frac{1}{2}, \sin^2 \epsilon y\right) & \text{when } n \text{ is even} \\ \cos^{\delta} \epsilon y \sin^{\delta} \epsilon y F\left(\delta + \frac{n+1}{2}, -\frac{n-1}{2}, \frac{3}{2}, \sin^2 \epsilon y\right) & \text{when } n \text{ is odd} \end{cases}$$

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

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

The eigenvalues of (3.13) are given by $\lambda_{n\epsilon}^{\pm}$ and $\lambda_{n\epsilon}^0$, $n = 0, 1, 2, \dots$ the eigenvalues $\lambda_{n\epsilon}^{\pm}$ have multiplicity two. Moreover as $\epsilon \rightarrow 0$ $\lambda_{n\epsilon}^0$, $\lambda_{n\epsilon}^{\pm}$ approaches the eigenvalues (2.24), (2.25), (2.26) of the three harmonic oscillators considered before.

The eigenfunctions of (3.14) satisfy $\frac{d}{dy} \left(\frac{\lambda_{n\epsilon}^{\pm}}{\beta \epsilon} \right) = \frac{d}{dy} \left(\frac{\lambda_{n\epsilon}^0}{\beta \epsilon} \right) = 0$ so that corresponding eigenfunctions of (3.13) can be obtained extending $\hat{\lambda}_{n\epsilon}^{\pm}(y)$ with zero for $y \notin A_{\pm}$. 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 $\lambda_{n\epsilon}^{\pm}$ of (3.13) have multiplicity two, the eigenfunction of (3.13) can be chosen to be even or odd.

Let $C_0^{\infty}(\mathbb{R} - \{\pm \frac{\alpha \sqrt{6}}{\beta \epsilon}\}) = \{f | f \text{ is } C^{\infty} \text{ and of compact support and is zero in a neighborhood of } y = \pm \frac{\alpha \sqrt{6}}{\beta \epsilon}\}$. We have:

Theorem 3.5. $M_{\beta \epsilon}^{(\pm)}$ is essentially self adjoint on $C_0^{\infty}(\mathbb{R} - \{\pm \frac{\alpha \sqrt{6}}{\beta \epsilon}\})$.

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

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

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:

$$(3.23) \quad U_{2\varepsilon} = \begin{cases} \frac{c_2^2}{4} \left[y - \eta_2 - \frac{1}{2v_2} \frac{1}{y - \eta_2} \right]^2 - c_2 & \text{when } y \in \bar{A}_+ \\ \frac{\bar{V}_0}{\cos^2 \bar{\varepsilon}(y - y_1)} - \bar{V}_0 - c_1 & \text{when } y \in \bar{A}_0 \\ \frac{c_2^2}{4} \left[y - (2y_1 - \eta_2) - \frac{1}{2v_1} \frac{1}{y - (2y_1 - \eta_2)} \right]^2 - c & \text{when } y \in \bar{A}_- \end{cases}$$

(see Fig. 7)

where

$$(3.24) \quad \frac{1}{2v_2} = (y_2 - \eta_2)^2$$

$$(3.25) \quad \frac{1}{2v_1} = (2y_1 - \eta_2)^2$$

$$(3.26) \quad \bar{\varepsilon} = \frac{\pi}{2} \frac{1}{\eta_2 - y_1}$$

$$(3.27) \quad \bar{V}_0 = \frac{c_1^2}{\bar{\varepsilon}^2}$$

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

$$(4.57) \quad |\hat{V}_{1\epsilon}'''(y)| \leq \frac{1}{4\nu^2} \frac{96\alpha^4}{\left|y - \frac{\alpha\sqrt{6}}{3\epsilon}\right|^5} \leq 96\alpha^4 \frac{\sqrt{2\nu}}{|1 - \bar{\eta}_2\sqrt{2\nu}|^5}$$

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

$$(4.58) \quad |F_\epsilon(y)| \leq \text{constant} \quad \epsilon^{1-3\delta_1}$$

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

Reasoning in the same way it can be shown that:

$$(4.59) \quad |F_\epsilon(y)| \leq \text{constant} \quad \epsilon^{1-3\delta_1}$$

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

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

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

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

and for $0 < \epsilon < \epsilon_0$

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

and

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

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

and

$$(4.53) \quad |\hat{V}_{1\varepsilon}''(y)| = 32\alpha^4 \beta |\cos^{-5}\delta y| |\sin\delta y| (2 + \sin^2\delta y) \\ \leq \frac{\varepsilon \cdot 24\sqrt{6} \cdot \alpha^3 \pi}{|\cos^5(\frac{\pi}{2} - \frac{\beta}{\alpha\sqrt{6}} \varepsilon^{1-\delta_1})|}$$

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

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

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

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

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

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

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

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

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

and

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

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

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

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

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

for $0 < \varepsilon < \varepsilon_0$.

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

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

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

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

with ξ an intermediate point in the interval $(0, y)$.

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

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

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

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

$\bar{\eta}_2(\varepsilon)$ 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:

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

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

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

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

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

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

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

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

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

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

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

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:

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

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

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

Proof: Note that (4.27), (4.28), (4.29) imply \hat{V}_ε , $\hat{V}_{1\varepsilon}$, $\hat{V}_{2\varepsilon} \geq \text{constant} > 0$ so that \hat{V}_ε^{-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 530 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 $\chi_{I_1^{(\varepsilon)} \cup I_2^{(\varepsilon)}}$.

$$\begin{aligned}
(4.26) \quad & (1 - \chi_{I_1(\epsilon) \cup I_2(\epsilon)}) \{ (1 - \bar{\beta}) V_{2\epsilon}^2 + 2z V_{2\epsilon} + z^2 - V_{2\epsilon}'' \} + \\
& + \chi_{I_1(\epsilon) \cup I_2(\epsilon)} \{ (1 - \bar{\beta}) V_{2\epsilon}^2(\bar{\eta}_1) + 2z V_{2\epsilon}(\bar{\eta}_1) + z^2 \} + \\
& + \bar{c}_1 \{ \delta(y - \bar{\eta}_1) + \delta(y + \bar{\eta}_1) \} + \bar{c}_2 \{ \delta(y - \frac{\alpha\sqrt{2}}{\epsilon} + \bar{\eta}_2) + \delta(y + \frac{\alpha\sqrt{2}}{\epsilon} - \bar{\eta}_2) \} \geq 0.
\end{aligned}$$

In fact for $z > z_0 > 0$, $0 < \epsilon < \epsilon_0$ we have

$$\{ (1 - \bar{\beta}) V_{2\epsilon}^2 + 2z V_{2\epsilon} + z^2 - V_{2\epsilon}'' \} \geq 0. \text{ Moreover}$$

$$\{ (1 - \bar{\beta}) V_{2\epsilon}^2(\bar{\eta}_1) + 2z V_{2\epsilon}(\bar{\eta}_1) + z^2 \} \geq 0 \text{ and } \bar{c}_1 \geq 0, \bar{c}_2 \geq 0.$$

The estimate (4.2) is established.

Let \hat{c} be a constant such that

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

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

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

We define

$$(4.30) \quad \hat{M}_\epsilon = - \frac{d^2}{dy^2} + \hat{V}_\epsilon = M_\epsilon + \hat{c}$$

$$(4.31) \quad \hat{M}_\epsilon^{(1)} = - \frac{d^2}{dy^2} + \hat{V}_{1\epsilon} = M_\epsilon^{(1)} + \hat{c}$$

$$(4.32) \quad \hat{M}_\epsilon^{(2)} = - \frac{d^2}{dy^2} + \hat{V}_{2\epsilon} = M_\epsilon^{(2)} + \hat{c}$$

Theorem 4.2. There exist $z_0 > 0$ and $\epsilon_0 > 0$ such that for $z \geq z_0$ and $0 < \epsilon < \epsilon_0$ 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 $C_0^\infty(\mathbb{R}) \times C_0^\infty(\mathbb{R})$

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

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

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

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

We have

$$(4.23) \quad V_{1\varepsilon}' = V_{2\varepsilon}' (1 - \chi_{I_1^{(\varepsilon)} \cup I_2^{(\varepsilon)}}) - \bar{c}_1 [\delta(y - \bar{\eta}_1) + \delta(y + \bar{\eta}_1)] \\ - \bar{c}_2 [\delta(y - \frac{\alpha\sqrt{2}}{\varepsilon} + \bar{\eta}_2) + \delta(y + \frac{\alpha\sqrt{2}}{\varepsilon} - \bar{\eta}_2)]$$

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

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

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

are the absolute values of the jumps at $y = \pm \bar{\eta}_1$ and $y = \pm (\frac{\alpha\sqrt{2}}{\varepsilon} - \bar{\eta}_2)$ 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 $\varepsilon \rightarrow 0$. The last inequality in (4.16) holds $\forall z > 0$, $0 < \varepsilon < \frac{2}{3} \frac{\alpha^2}{\pi} (6(1+\beta))^{1/4}$.

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

$$(4.17) \quad z^2 + 2 \left[4\alpha^4 \left(y - \frac{\alpha\sqrt{6}}{3\varepsilon} - \frac{1}{2\nu} \frac{1}{y - \frac{\alpha\sqrt{6}}{3\varepsilon}} \right)^2 - 4\alpha^2 \right] z + \\ + (1-\beta) \left[4\alpha^4 \left(y - \frac{\alpha\sqrt{6}}{3\varepsilon} - \frac{1}{2\nu} \frac{1}{y - \frac{\alpha\sqrt{6}}{3\varepsilon}} \right)^2 - 4\alpha^2 \right]^2 + \\ - 8\alpha^4 \left(1 + \frac{3}{4\nu^2} \frac{1}{\left(y - \frac{\alpha\sqrt{6}}{3\varepsilon} \right)^4} \right) \geq 0$$

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

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

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

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

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

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

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

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 $C_0^\infty(\mathbb{R} - \{\pm \frac{\alpha\sqrt{6}}{3\varepsilon}\}) \times C_0^\infty(\mathbb{R} - \{\pm \frac{\alpha\sqrt{6}}{3\varepsilon}\})$

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

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

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

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

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

when $|y| \leq \frac{\pi}{4\bar{\beta}}$ 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

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

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

When $\frac{\pi}{4\bar{\beta}} \leq |y| < \frac{\pi}{2\bar{\beta}}$ 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

$$(4.16) \quad z^2 + 4\alpha^2 z + \frac{1}{\cos^4 \beta y} [(1 - \bar{\beta}) V_0^2 \sin^4 \beta y - 8\alpha^4 (2 \sin^2 \beta y + 1)] \geq \\ \geq z^2 + 4\alpha^2 z + 4 \left[\frac{1 - \bar{\beta}}{4} V_0^2 - 24\alpha^4 \right] \geq 0$$

$$(4.7) \quad F_1(y^2) = 2zV_\epsilon + z^2 - V'_\epsilon \geq 0 \quad y \in \mathbb{R}$$

Let us define

$$\begin{aligned} t &= y^2 \\ A &= 2z\epsilon^4 \\ B &= 8z\alpha^2\epsilon^2 + 30\epsilon^4 \\ C &= 2z(4\alpha^4 - 3\epsilon^2) + 48\alpha^2\epsilon^2 \\ D &= 4z\alpha^2 + z^2 - 8\alpha^4 + 6\epsilon^2 \end{aligned}$$

A simple computation shows that

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

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

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

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

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

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 $\forall t \geq 0$

$$(4.11) \quad \begin{aligned} F_1(t) &\geq \frac{B}{A} (At_0^2 - Bt_0 + C) + D = \\ &= z^2 - 20\alpha^2z - 56\alpha^4 - 84\epsilon^2 - 630 \frac{\alpha^2\epsilon^2}{z} - \frac{3375}{2} \frac{\epsilon^4}{z^2} \end{aligned}$$

§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 $z \geq z_0$ and $0 < \varepsilon < \varepsilon_0$ we have

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

$$(4.2) \quad (M_\varepsilon^{(1)} + z)^2 \geq \bar{B}V_{1\varepsilon}^2 \quad \text{on } C_0^\infty(\mathbb{R}) \times C_0^\infty(\mathbb{R})$$

$$(4.3) \quad (M_\varepsilon^{(2)} + z)^2 \geq \bar{B}V_{2\varepsilon}^2 \quad \text{on } C_0^\infty(\mathbb{R} - \{\pm \frac{\alpha\sqrt{6}}{3\varepsilon}\}) \times C_0^\infty(\mathbb{R} - \{\pm \frac{\alpha\sqrt{6}}{3\varepsilon}\})$$

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:

$$(4.4) \quad \begin{aligned} (M_\varepsilon + z)^2 &= (p^2 + V_\varepsilon + z)^2 = \\ &= p^4 + V_\varepsilon^2 + 2zV_\varepsilon + z^2 + 2p(V_\varepsilon + z)p - V_\varepsilon'' \end{aligned}$$

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

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

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

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

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

Proceeding as before let us now consider

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

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

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 $\varepsilon \rightarrow 0$ the eigenvalues of $N_{\varepsilon}^{(2)}$ approach the eigenvalues (2.30), (2.31), (2.32) of the three harmonic oscillators considered before.

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

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 [16].

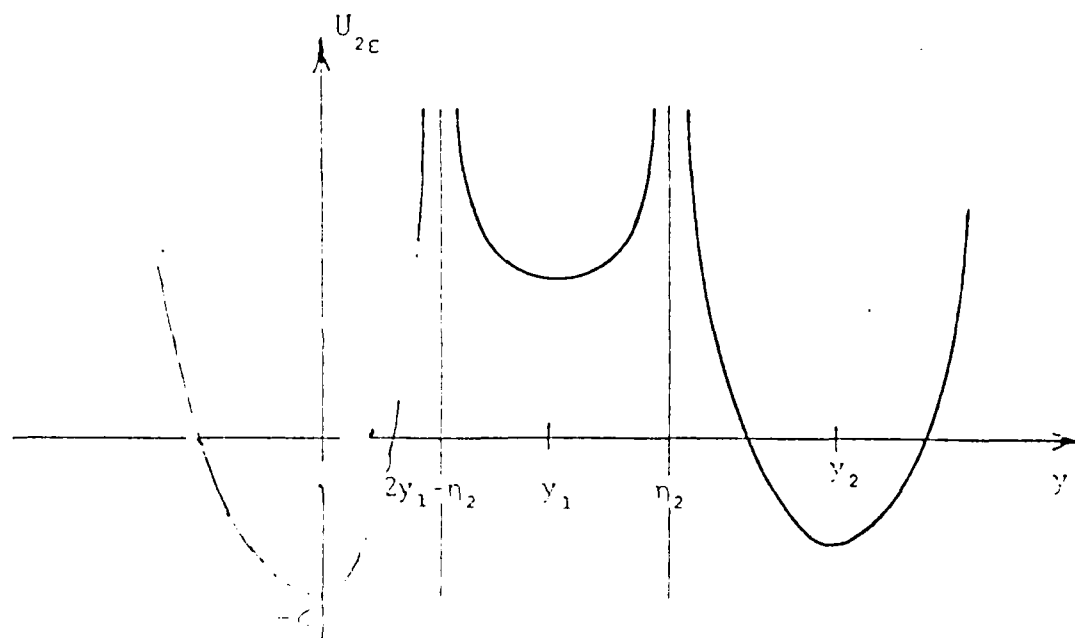


Fig. 7

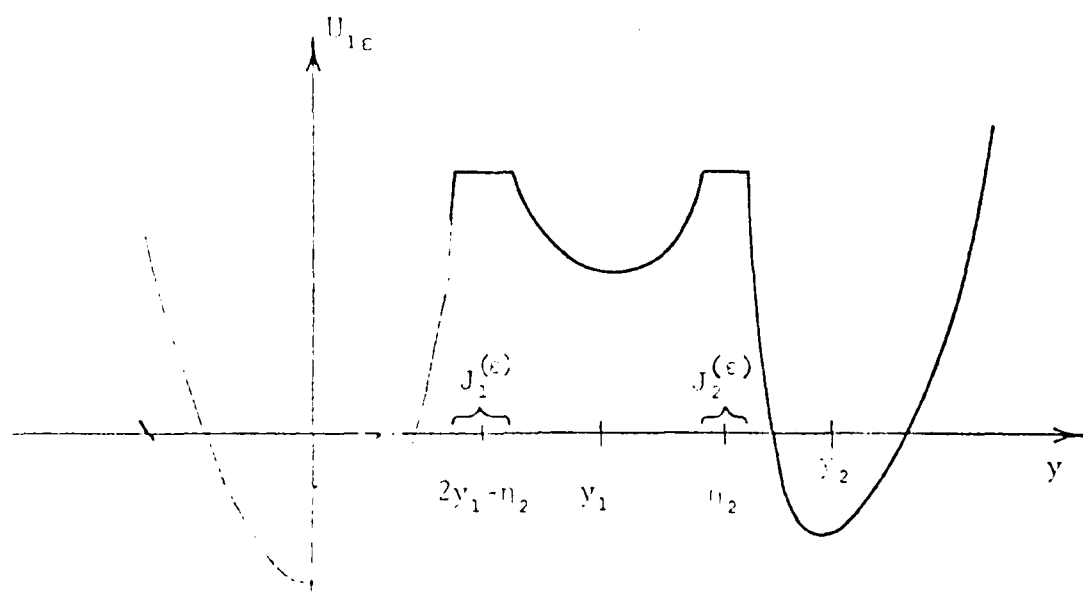


Fig. 8

is easy to check by explicit computation that $2y_1 - \eta_2 > 0$ so that the function $U_{2\varepsilon}$ (Fig. 7) as $\varepsilon \rightarrow 0$ 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

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

two intervals such that

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

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

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

and $\bar{J}_1^{(\varepsilon)} \cap \bar{J}_2^{(\varepsilon)} = \{\emptyset\}$, 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

$$(3.30) \quad \lim_{\varepsilon \rightarrow 0} U_{2\varepsilon}(y_1 + \mu_2(\varepsilon)) = \infty.$$

Let us define

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

(see Fig. 8)

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

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

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

where

$$(4.65) \quad m(\varepsilon) = \min_{y \in I_1^{(\varepsilon)} \cup I_2^{(\varepsilon)}} \hat{V}_{\varepsilon}(y) = \min_{y \in I_1^{(\varepsilon)}} \hat{V}_{\varepsilon}(y) = \\ = \min\{\hat{V}_{\varepsilon}(\bar{\eta}_1(\varepsilon)), \hat{V}_{\varepsilon}\left(\frac{\alpha\sqrt{\varepsilon}}{\varepsilon} - \bar{\eta}_2(\varepsilon)\right)\}$$

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

An elementary computation now shows that

$$(4.66) \quad |\hat{V}_{\varepsilon}^{-1}(\hat{V}_{\varepsilon} - \hat{V}_{1\varepsilon})| < \text{constant}, \quad \text{for } 0 < \varepsilon < \varepsilon_0 \quad \text{when } y \in I_1^{(\varepsilon)} \cup I_2^{(\varepsilon)}.$$

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

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

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

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

Moreover

$$(4.69) \quad V_{\varepsilon}(y) = -4\alpha^2 - 6\sqrt{2} \alpha\varepsilon \left(y - \frac{\alpha\sqrt{2}}{\varepsilon}\right) + (16\alpha^4 - 3\varepsilon^2) \left(y - \frac{\alpha\sqrt{2}}{\varepsilon}\right)^2 + \\ + 24\sqrt{2} \alpha^3\varepsilon \left(y - \frac{\alpha\sqrt{2}}{\varepsilon}\right)^3 + 26 \alpha^2\varepsilon^2 \left(y - \frac{\alpha\sqrt{2}}{\varepsilon}\right)^4 + \\ + 6\sqrt{2} \alpha^3 \left(y - \frac{\alpha\sqrt{2}}{\varepsilon}\right)^5 + \varepsilon^4 \left(y - \frac{\alpha\sqrt{2}}{\varepsilon}\right)^6$$

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

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

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

$$(4.71) \quad |\hat{V}_{\varepsilon}^{-1}(\hat{V}_{\varepsilon} - \hat{V}_{1\varepsilon})| \leq \text{constant} \quad \text{when } 0 < \varepsilon < \varepsilon_0, y > \frac{\alpha\sqrt{2}}{\varepsilon} + \bar{\eta}_2(\varepsilon).$$

Reasoning in the same way it can be shown that

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

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

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

and

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

From (4.39) it follows that:

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

Moreover

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

and

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

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

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

Reasoning in the same way it can be shown that

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

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:

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

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

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

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

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

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

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

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

A simple computation shows that

$$(4.85) \quad F_z(y) = 2a^2\varepsilon^4zy^6 + \frac{6ab}{\sqrt{2}}\varepsilon^3zy^5 + \left\{2z\left(\frac{9}{8}b^2 + 2ac\right)\varepsilon^2 - 30a^2\varepsilon^4\right\}y^4 + \\ + \left\{\frac{6}{\sqrt{2}}bc\varepsilon z - \frac{60ab}{\sqrt{2}}\varepsilon^3\right\}y^3 + \left\{2z(c^2 - 3ac^2) - 12\left(\frac{9}{8}b^2 + 2ac\right)\varepsilon^2\right\}y^2 + \\ - \left\{\frac{6b}{\sqrt{2}}\varepsilon z + \frac{18bc}{\sqrt{2}}\varepsilon\right\}y - 2zc - 2c^2 + 6a\varepsilon^2 + z^2$$

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

$$(4.86) \quad F_2\left(\frac{t}{\epsilon}\right) = 2z\left\{\left[\frac{1}{\epsilon^2} \frac{1}{4} t^2 (2at^2 + \frac{3b}{\sqrt{z}} t + 2c)^2 - (3at^2 + \frac{3b}{\sqrt{z}} t + c) + \frac{z}{2}\right] + \right. \\ \left. - \frac{1}{2z} \left[30a^2 t^4 + \frac{60ab}{\sqrt{z}} t^3 + 12\left(\frac{9}{8}b^2 + 2ac\right)t^2 + \frac{18}{\sqrt{z}} bct + 2c^2 - 6a\epsilon^2\right]\right\}$$

For $z > z_0$ and $0 < \epsilon < \epsilon_0$ 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

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

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

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

We define

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

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

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

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

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

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

$$(4.95) \quad (\hat{N}_\varepsilon^{(2)} + z)^2 \geq \bar{\beta} \hat{U}_{2\varepsilon}^2 \quad \text{on } C_0^\infty(\mathbb{R} - \{2y_1 - n_2\}) \times C_0^\infty(\mathbb{R} - \{2y_1 - n_2\} - \{n_2\})$$

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

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:

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

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

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

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

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 $\mathbb{R} \setminus U_*^{(\varepsilon)}$ 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

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

$\mu_2(\varepsilon)$ 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:

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

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

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

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

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

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

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 $\varepsilon \rightarrow 0$ of eigenvalues and eigenvectors of $M_\varepsilon, N_\varepsilon$.

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

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

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

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

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

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

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

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 $M_t^{(2)}$ and the resolvent of N_ε converges to the resolvent of $N_t^{(2)}$ as $\varepsilon \rightarrow 0$. In section 3 we have

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

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

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

$$(5.7) \quad \lambda_{n\epsilon}^\pm = 8\alpha^2 n + O(\epsilon^2) \quad n = 0, 1, 2, \dots$$

$$(5.8) \quad \lambda_{n\epsilon}^0 = 4\alpha^2(n+1) + O(\epsilon^2) \quad n = 0, 1, 2, \dots$$

(see Fig. 9).

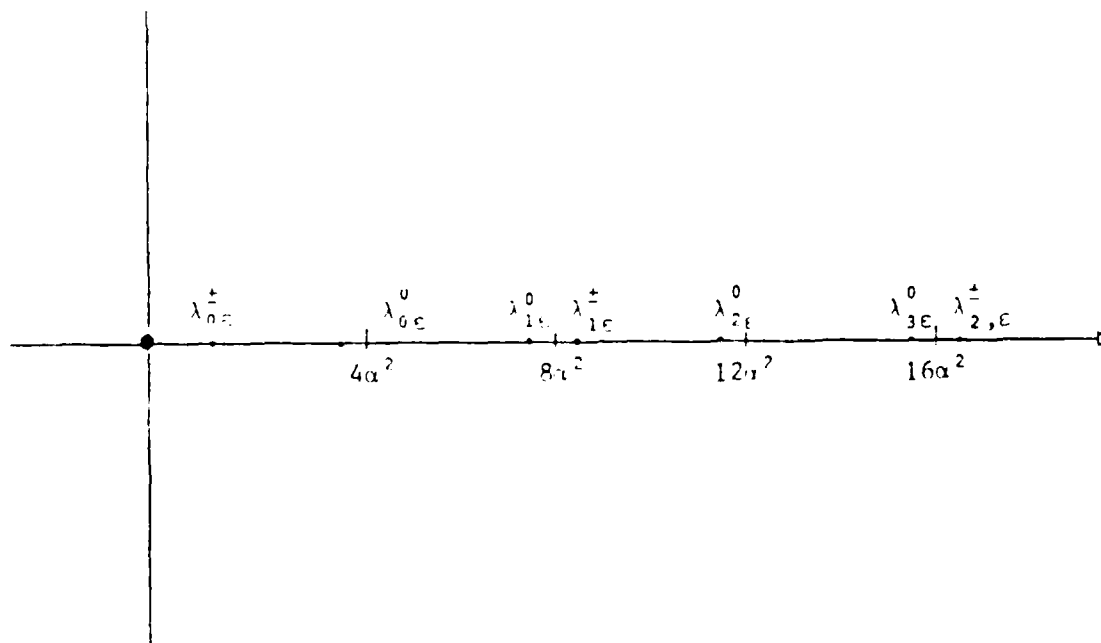


Fig. 9

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

let

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

and

$$(5.10) \quad D_k = \{z \mid |z - 4\alpha^2 k| \leq r\} \quad k = 0, 1, 2, \dots$$

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

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

then

$$(5.12) \quad P_\epsilon^{(2)}(D_0) = P_\epsilon^{(2)}(\{\lambda_{0\epsilon}^\pm\}) \quad \text{for } \epsilon \text{ small enough}$$

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

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

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

$$(5.14) \quad (z - M_\epsilon)^{-1} \text{ exists}$$

$$(5.15) \quad \sup_{z \in C_k(r)} \| (z - M_\varepsilon)^{-1} - (z - M_\varepsilon^{(2)})^{-1} \| \leq \text{constant} \varepsilon^{\frac{k+1}{2}}$$

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 $k = 0, 1, 2, \dots$ we have

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

Moreover for all ε sufficiently small M_ε possesses:

(i) two distinct eigenvalues $\mu_0(\varepsilon) \equiv 0$, $\mu_0'(\varepsilon) > 0$ such that:

$$(5.16) \quad \lim_{\varepsilon \rightarrow 0} \mu_0'(\varepsilon) = \mu_0(\varepsilon) \equiv 0$$

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

$$(5.17) \quad \lim_{\varepsilon \rightarrow 0} \mu_k(\varepsilon) = 4\alpha^2 k \quad k = 1, 3, \dots$$

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

$$(5.18) \quad \lim_{\varepsilon \rightarrow 0} \mu_k(\varepsilon) = \lim_{\varepsilon \rightarrow 0} \mu_k'(\varepsilon) = \lim_{\varepsilon \rightarrow 0} \mu_k''(\varepsilon) = 4\alpha^2 k \quad k = 2, 4, \dots$$

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

$$(5.19) \quad \| P_\varepsilon(D_k) - P_\varepsilon^{(2)}(D_k) \| = \left\| \frac{1}{2\pi i} \oint_{C_k(r)} [(z - M_\varepsilon)^{-1} - (z - M_\varepsilon^{(2)})^{-1}] dz \right\| \\ \leq \text{constant} \quad r = \frac{k+1}{2}$$

So that for ε sufficiently small

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

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

$$(5.21) \quad \lim_{\varepsilon \rightarrow 0} -\lambda_1(\varepsilon) = 0$$

$$(5.22) \quad \lim_{\varepsilon \rightarrow 0} -\lambda_{2+4n}(\varepsilon) = 4\alpha^2(2n+1) \quad n = 0, 1, 2, \dots$$

$$(5.23) \quad \lim_{\varepsilon \rightarrow 0} -\lambda_{3+4n}(\varepsilon) = \lim_{\varepsilon \rightarrow 0} -\lambda_{4+4n}(\varepsilon) = \lim_{\varepsilon \rightarrow 0} -\lambda_{5+4n}(\varepsilon) = 8\alpha^2(n+1) \\ n = 0, 1, 2, \dots$$

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

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

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 $\varepsilon \rightarrow 0$ are given by:

$$(5.24) \quad -\bar{\lambda}_{n\varepsilon}^{(1)} = c(2n+1) - c + o(\varepsilon^2) \quad n = 0, 1, 2, \dots$$

$$(5.25) \quad -\bar{\lambda}_{n\varepsilon}^{(2)} = |c_1|(2n+1) - c_1 + o(\varepsilon^2) \quad n = 0, 1, 2, \dots$$

$$(5.26) \quad -\bar{\lambda}_{n\varepsilon}^{(3)} = c_2(2n+1) - c_2 + o(\varepsilon^2) \quad n = 0, 1, 2, \dots$$

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 $-\bar{\lambda}_n \leq -\bar{\lambda}_{n+1}$ $n = 0, 1, \dots$. 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_3 we will have $-\bar{\lambda}_0 = -\bar{\lambda}_1 = 0$.

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

Then

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

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

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

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

That is, there are eigenvalues of N_ε that remain isolated when $\varepsilon \rightarrow 0$ and eigenvalues that have asymptotic multiplicity three when $\varepsilon \rightarrow 0$. 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.

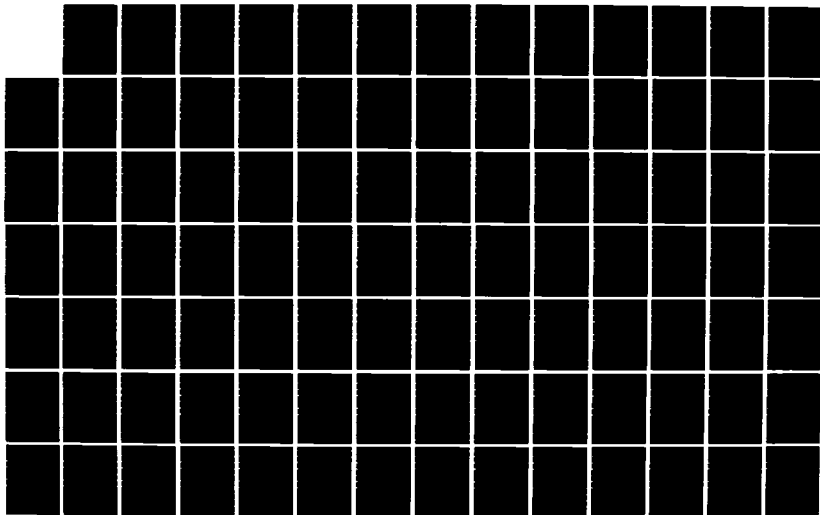
2/8

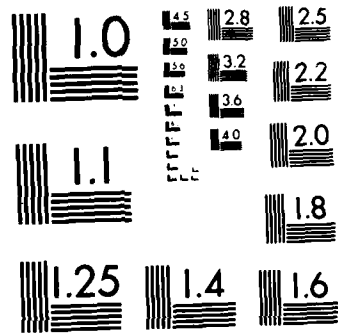
UNCLASSIFIED

DEC 84 DAJA37-81-C-0740

F/G 12/1

NL





MICROCOPY RESOLUTION TEST CHART
NBS 1963-A

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

In section 5 it has been shown that

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

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

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:

$$(6.3) \quad v_0(y) = d_\varepsilon^{1/2} e^{-f_1 \varepsilon / 2}$$

and

$$(6.4) \quad \bar{v}_0(y) = \bar{d}_\varepsilon^{1/2} e^{-\bar{f}_2 \varepsilon / 2}$$

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

$$(6.5) \quad d_\varepsilon = \left(\int_{-\infty}^{+\infty} e^{-f_1 \varepsilon / 2} dy \right)^{-1} = \frac{\varepsilon}{\sqrt{2}} c_\varepsilon$$

$$(6.6) \quad \bar{d}_\varepsilon = \left(\int_{-\infty}^{+\infty} e^{-\bar{f}_2 \varepsilon / 2} dy \right)^{-1} = \frac{\varepsilon}{\sqrt{2}} \bar{c}_\varepsilon$$

are normalization constants such that $\|v_0\|_{L^2(\mathbb{R})} = \|\bar{v}_0\|_{L^2(\mathbb{R})} = 1$ and $c_\varepsilon, \bar{c}_\varepsilon$ 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 $\varepsilon \rightarrow 0$ 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 $\varepsilon \rightarrow 0$ we have

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

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

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

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

$$(6.9) \quad g = uv_0$$

where $u(y) = -u(-y)$ is an odd function such that $u \in L^\infty(\mathbb{R})$ and

$\frac{du}{dy} \in L^1(\mathbb{R})$ 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

$$\begin{aligned}
 (6.10) \quad \langle g, M_\varepsilon g \rangle_{L^2(\mathbb{R})} &= \int_{-\infty}^{+\infty} uv_0 \left(-\frac{d^2}{dy^2} + V_\varepsilon \right) uv_0 dy = \\
 &= \int_{-\infty}^{+\infty} \left\{ \left[\frac{d}{dy} (uv_0) \right]^2 + V_\varepsilon u^2 v_0^2 \right\} dy = \\
 &= \int_{-\infty}^{+\infty} \left\{ \left(\frac{du}{dy} \right)^2 v_0^2 + u^2 \left(\frac{dv_0}{dy} \right)^2 + 2u \frac{du}{dy} v_0 \frac{dv_0}{dy} + V_\varepsilon u^2 v_0^2 \right\} dy \\
 &= \int_{-\infty}^{+\infty} \left(\frac{du}{dy} \right)^2 v_0^2 dy
 \end{aligned}$$

Since

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

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

So that

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

Let us choose

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

equation (6.12) becomes

$$(6.13) \quad 0 < -\lambda_1(\epsilon) + \lambda_0(\epsilon) = -\lambda_1(\epsilon) \leq \frac{\int_{-1}^1 v_0^2 dy}{\int_{-\infty}^{+\infty} u^2 v_0^2 dy}$$

Moreover

$$(6.14) \quad \int_{-1}^1 v_0^2 dy = d_\epsilon \int_{-1}^{+1} e^{-f_{1\epsilon}(y)} dy = \frac{\epsilon}{\sqrt{2}} c_\epsilon \int_{-1}^1 e^{-f_{1\epsilon}(\epsilon)} dy \\ \leq \frac{\epsilon}{\sqrt{2}} c_\epsilon 2e^{-f_{1\epsilon}(1)} = \frac{\epsilon}{\sqrt{2}} c_\epsilon 2 e^{+2\alpha^2} e^{-\frac{\epsilon^2}{2}} e^{-\frac{2}{\epsilon^2}\alpha^4}$$

It can be easily shown that

$$(6.15) \quad \lim_{\epsilon \rightarrow 0} \epsilon c_\epsilon = \sqrt{\frac{2}{\pi}} \alpha$$

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

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

In fact in the sense of distribution

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

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 $\alpha^* = f_1(0) - f_1(\alpha)$ 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 $\varepsilon \rightarrow 0$ we have

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

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

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

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

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

where

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

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

Reasoning as in Theorem 6.1 it can be shown that:

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

Moreover

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

and

$$(6.24) \quad \lim_{\varepsilon \rightarrow 0} \int_{-\infty}^{+\infty} \bar{u}^2 \bar{v}_0^2 dy = 1$$

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

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

$$(7.1) \quad f'(x) = 0$$

has n roots z_1, z_2, \dots, z_n such that

$$(7.2) \quad f''(z_i) = \alpha_i \neq 0 \quad i = 1, 2, \dots, n$$

that is z_1, z_2, \dots, z_n are non-degenerate minimizers or maximizers of f .

Let

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

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

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

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

Let $y_i = \frac{\sqrt{\varepsilon}}{\varepsilon} z_i \quad i = 1, 2, \dots, n$ a straightforward computation shows that as $\varepsilon \rightarrow 0$ $W_\varepsilon(y)$ 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 $\lambda_k^i = \frac{1}{2}\alpha_i^2(2k+1) - \frac{1}{2}\alpha_i$, $i = 1, 2, \dots, n$ and $k = 0, 1, 2, \dots$.

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

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 Applied 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. Chandrasekhar: "Stochastic problems in physics and astronomy" in *Selected Papers on Noise and Stochastic 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
 $(x,y) = \pm(0, 14.94511)$, 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
 $(x,y) = \pm(0, 26.58678)$, 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:

$$f(x,y) = g(x,y) h(x,y)$$

$$\text{with } g(x,y) = 1 + u^2 (36 - 20u + 3u^2)$$

$$h(x,y) = 30 + v^2 (18 - 16v + 3v^2)$$

$$\text{where } u = x + y + 1 \text{ and } v = 28 - 3v$$

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
 $(x, y) = \pm(0, 2.60891)$, 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
 $(x, y) = \pm(0, 4.70174)$, 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
 $(x, y) = \pm(0, 8.39401)$, 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,
aving 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: $\beta = 0$

$$a = -1.4251284283197609708, \quad b = -0.80032110047197312466$$

c) dimension: $N = 2$ d) region: $D = \{ |x| \leq 10, |y| \leq 10 \}$ 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

$$\begin{aligned} (x,y) &= (-7.08350, -7.70831) \\ (x,y) &= (-0.80032, -7.70831) \\ (x,y) &= (5.48286, -7.70831) \\ (x,y) &= (-7.70831, -7.08350) \\ (x,y) &= (-1.42513, -7.08350) \\ (x,y) &= (4.85805, -7.08350) \\ (x,y) &= (-7.08350, -1.42513) \\ (x,y) &= (-0.80032, -1.42513) \\ (x,y) &= (5.48286, -1.42513) \\ (x,y) &= (-7.70831, -0.80032) \\ (x,y) &= (-1.42513, -0.80032) \\ (x,y) &= (4.85805, -0.80032) \\ (x,y) &= (7.08350, 4.85805) \\ (x,y) &= (-8.00320, 4.85805) \\ (x,y) &= (5.48286, 4.85805) \\ (x,y) &= (-7.70831, 5.48286) \\ (x,y) &= (-1.42513, 5.48286) \\ (x,y) &= (4.85805, 5.48286), \text{ where } f = -186.73091 \end{aligned}$$

and 742 other local minima.

g) initial point: $(x_0, y_0) = (0, 0)$

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

$$(x,y) = (0, 0), \text{ where } f = 0$$

and six other local minima at:

$$(x,y) = \pm(2.98978, 0), \text{ where } f = 0.46981$$

$$(x,y) = \pm(5.96370, 0), \text{ where } f = 1.97693$$

$$(x,y) = \pm(8.87846, 0), \text{ where } f = 4.21128$$

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:

$$f(x,y) = (4 - 2.1x^2 + x^4/3)x^2 + xy + (-4 + 4y^2)y^2$$

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

$$(x,y) = \pm(-0.089842, 0.71266), \text{ where } f = -1.03163$$

and four other local minima, at

$$(x,y) = \pm(-1.70361, 0.79608), \text{ where } f = -0.21546, \text{ and}$$

$$(x,y) = \pm(1.60710, 0.56865), \text{ where } f = 2.10425$$

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: $D = (|x| \leq 10)$

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:

$$f(x, y) = x^4/4 - x^2/2 + ax + y^2/2$$

b) parameter values: $a = 0.1$

c) dimension: $N = 2$

d) region: $D = (|x| \leq 10, |y| \leq 10)$

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
 $x = \pm 3$, 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. Hillstom, "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 operation 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, X0, 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)

X0 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

$$XMIN(I) \leq X(I) \leq XMAX(I), \quad I = 1, \dots, N$$

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

$$D = (x_{\min_i} \leq x_i \leq x_{\max_i}, \quad i = 1, \dots, N)$$

- 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 $a > 0, b > 0$)

$$u(x, a, b, m) = \begin{cases} b (|x| - a)^m & (|x| > a) \\ 0 & (|x| \leq a) \end{cases}$$

- 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: $(x_0, y_0) = (1, 1)$
- 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: $(x_0, y_0) = (2.5, 7.5)$
- 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

$$A = [a_{ij}] = \begin{array}{|cccc|} \hline 4 & 4 & 4 & 4 \\ \hline 1 & 1 & 1 & 1 \\ \hline 8 & 8 & 8 & 8 \\ \hline 6 & 6 & 6 & 6 \\ \hline 3 & 7 & 3 & 7 \\ \hline 2 & 9 & 2 & 9 \\ \hline 5 & 5 & 3 & 3 \\ \hline 8 & 1 & 8 & 1 \\ \hline 6 & 2 & 6 & 2 \\ \hline 7 & 3.6 & 7 & 3.6 \\ \hline \end{array}$$

b) parameter values: $M = 5$

c) dimension: $N = 4$

d) region: $D = \{ 0 \leq x_j \leq 10, \quad j = 1, \dots, N \}$

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: $x_{oj} = 9, j = 1, \dots, N$
- 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: $D = \{ 0 \leq x_j \leq 1, \quad j = 1, \dots, N \}$

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: $x_{0j} = 0.5, \quad j = 1, \dots, N$

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: $M = 4$

$$A = [a_{ij}] = \begin{bmatrix} 10 & 3 & 17 & 3.5 & 1.7 & 8 \\ .05 & 10 & 17 & .1 & 8 & 14 \\ 3 & 3.5 & 1.7 & 10 & 17 & 8 \\ 17 & 8 & .05 & 10 & 0.1 & 14 \end{bmatrix}$$

$$P = [p_{ij}] = \begin{bmatrix} 0.1312 & 0.1696 & 0.5569 & 0.0124 & 0.8283 & 0.5886 \\ 0.2329 & 0.4135 & 0.8307 & 0.3736 & 0.1004 & 0.9991 \\ 0.2348 & 0.1451 & 0.3522 & 0.2883 & 0.3047 & 0.6650 \\ 0.4047 & 0.8828 & 0.8732 & 0.5743 & 0.1091 & 0.0381 \end{bmatrix}$$

c) dimension: $N = 6$ d) region: $D = \{ 0 \leq x_j \leq 1, j = 1, \dots, N \}$

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: $x_{0j} = 0.5, j = 1, \dots, N$

h) source: Appendix 1 in ref. [3]

i) notes: none.

Problem 23

Penalized Levy-Montalvo function, type 1, $N = 2$

a) function:

$$f(x) = (\pi/N) (10 \sin^2(\pi y_1) + (y_N - 1)^2 + \\ + \sum_{j=1}^{N-1} (y_j - 1)^2 [1 + 10 \sin^2(\pi y_{j+1})])$$

$$\text{where } y_j = 1 + (x_j - 1)/4, \quad j = 1, \dots, N$$

b) parameter values: none

c) dimension: $N = 2$ d) region: $D = \{ -10 \leq x_j \leq 10, \quad j = 1, \dots, N \}$

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

$$x_j = 1, \quad j = 1, \dots, N, \quad \text{where } f = 0$$

and a number of local minima of the order of 5^N g) initial point: $x_{0j} = 0, \quad j = 1, \dots, N$

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

$$y_i = x_i, \quad i = 1, \dots, N$$

- 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:

$$f(x) = 0.1 \left(\sin^2(3\pi x_1) \right. \\
\left. + (x_N - 1)^2 [1 + \sin^2(2\pi x_N)] + \right. \\
\left. + \sum_{i=1}^{N-1} (x_i - 1)^2 [1 + \sin^2(3\pi x_{i+1})] \right)$$

- 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
 $x_j = 1, \quad j = 1, \dots, N, \quad \text{where } f = 0$
 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: $D = \{ -5 \leq x_i \leq 5, \quad i = 1, \dots, N \}$
 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: $D = \{ -20000 \leq x_j \leq 10000, \quad j = 1, \dots, N \}$
 e) penalization: none
 f) minima in the region D : a single global minimum at
 $x_j = 0, \quad j = 1, \dots, N, \quad \text{where } f = 0$
 g) initial point: $x_{0j} = 1000, \quad j = 1, \dots, N$
 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) functions:

$$f(x) = \|x\|^2 - (C + h) g(x)$$

where

$$g(x) = \begin{cases} \exp(-s / (b^2 - s)) & (s < b^2) \\ 0 & (s \geq b^2) \end{cases}$$

$$s = \|x - c\|^2, \quad C = \|c\|^2$$

and

$$x = (x_1, x_2, \dots, x_N), \quad c = (a, 0, 0, \dots, 0)$$

b) parameter values: $a = 100$, $b = 1$, $h = 10$

c) dimensions: $N = 2$

d) regions:

$$D = \{ -1000 \leq x_j \leq 1000, \quad j = 1, \dots, N \}$$

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) sources: 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) functions: as in problem 36
- b) parameter values: $a = 10$, $b = 1$, $h = 10$
- c) dimensions: $N = 5$
- d) regions: $D = \{ -100 \leq x_j \leq 100, \quad j = 1, \dots, N \}$
- e) penalizations: 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 points: $x_0 = (0, 0, 0, 0, 10)$
- h), i): as in probl. 36.

APPENDIX 2

The FORTRAN listing

	N = 5	GLIP2860
	V0 = 1000.00	GLIP2870
	VMIN = -20000.00	GLIP2880
	VMAX = 10000.00	GLIP2890
	GO TO 900	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) = 0.00	GLIP2960
	X0(2) = 100.00	GLIP2970
	VMIN = -1000.00	GLIP2980
	VMAX = 1000.00	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) = 0.00	GLIP3060
	X0(2) = 0.00	GLIP3070
	X0(3) = 0.00	GLIP3080
	X0(4) = 0.00	GLIP3090
	X0(5) = 10.00	GLIP3100
	VMIN = -100.00	GLIP3110
	VMAX = 100.00	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	23 PENALIZED LEVY-MONTALVO FUNCTION, TYPE 2 (N = 10)		GLIP2310
C			GLIP2320
	260 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
	VO = 0.00		GLIP2550
	VMIN = -10.00		GLIP2560
	VMAX = 10.00		GLIP2570
	GO TO 300		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
	VO = 0.00		GLIP2780
	VMIN = -5.00		GLIP2790
	VMAX = 5.00		GLIP2800
	GO TO 300		GLIP2810
C			GLIP2820
C	35 A FUNCTION WITH A SINGLE UBOB-SHAPED MINIMUM (N = 5)		GLIP2830
C			GLIP2840
	350 CONTINUE		GLIP2850

C	19	PENALIZED SHEKEL FUNCTION, M = 7 (N = 4)	GLIP1720
C			GLIP1730
C	190	CONTINUE	GLIP1740
C			GLIP1750
C	20	PENALIZED SHEKEL FUNCTION, M = 10 (N = 4)	GLIP1760
C			GLIP1770
C	200	CONTINUE	GLIP1780
C		N = 4	GLIP1790
C		VC = 9.00	GLIP1800
C		VMIN = 0.00	GLIP1810
C		VMAX = 10.00	GLIP1820
C		GO TO 830	GLIP1830
C			GLIP1840
C	21	PENALIZED THREE-DIMENSIONAL HARTMAN FUNCTION (N = 3)	GLIP1850
C			GLIP1860
C	210	CONTINUE	GLIP1870
C		N = 3	GLIP1880
C		GO TO 225	GLIP1890
C			GLIP1900
C	22	PENALIZED SIX-DIMENSIONAL HARTMAN FUNCTION (N = 6)	GLIP1910
C			GLIP1920
C	220	CONTINUE	GLIP1930
C		N = 6	GLIP1940
C	225	CONTINUE	GLIP1950
C		VO = 0.500	GLIP1960
C		VMIN = 0.00	GLIP1970
C		VMAX = 1.00	GLIP1980
C		GO TO 800	GLIP1990
C			GLIP2000
C	23	PENALIZED LEVY-MONTALVO FUNCTION, TYPE 1 (N = 2)	GLIP2010
C			GLIP2020
C	230	CONTINUE	GLIP2030
C		N = 2	GLIP2040
C		GO TO 315	GLIP2050
C			GLIP2060
C	24	PENALIZED LEVY-MONTALVO FUNCTION, TYPE 1 (N = 3)	GLIP2070
C			GLIP2080
C	240	CONTINUE	GLIP2090
C		N = 3	GLIP2100
C		GO TO 315	GLIP2110
C			GLIP2120
C	25	PENALIZED LEVY-MONTALVO FUNCTION, TYPE 1 (N = 4)	GLIP2130
C			GLIP2140
C	250	CONTINUE	GLIP2150
C		N = 4	GLIP2160
C		GO TO 315	GLIP2170
C			GLIP2180
C	26	PENALIZED LEVY-MONTALVO FUNCTION, TYPE 1 (N = 5)	GLIP2190
C			GLIP2200
C	260	CONTINUE	GLIP2210
C		N = 5	GLIP2220
C		GO TO 315	GLIP2230
C			GLIP2240
C	27	PENALIZED LEVY-MONTALVO FUNCTION, TYPE 2 (N = 8)	GLIP2250
C			GLIP2260
C	270	CONTINUE	GLIP2270
C			GLIP2280

C	10	A FUNCTION WITH THREE ILL-CONDITIONED MINIMA, A=10 (N = 2)	GLIP1150
C			GLIP1160
C	110	CONTINUE	GLIP1170
C			GLIP1190
C	11	A FUNCTION WITH THREE ILL-CONDITIONED MINIMA, A=10**2 (N = 2)	GLIP1190
C			GLIP1200
C	111	CONTINUE	GLIP1210
C			GLIP1220
C	12	A FUNCTION WITH THREE ILL-CONDITIONED MINIMA, A=10**3 (N = 2)	GLIP1230
C			GLIP1240
C	121	CONTINUE	GLIP1250
C			GLIP1260
C	13	A FUNCTION WITH THREE ILL-CONDITIONED MINIMA, A=10**4 (N = 2)	GLIP1270
C			GLIP1280
C	131	CONTINUE	GLIP1290
C			GLIP1300
C	14	A FUNCTION WITH THREE ILL-CONDITIONED MINIMA, A=10**5 (N = 2)	GLIP1310
C			GLIP1320
C	141	CONTINUE	GLIP1330
C			GLIP1340
C	15	A FUNCTION WITH THREE ILL-CONDITIONED MINIMA, A=10**6 (N = 2)	GLIP1350
C			GLIP1360
C	151	CONTINUE	GLIP1370
C		N = 2	GLIP1380
C		X0(1) = 0.00	GLIP1390
C		X0(2) = 0.00	GLIP1400
C		XMIN(1) = -10.00	GLIP1410
C		XMAX(1) = 10.00	GLIP1420
C		XMIN(2) = -100.00	GLIP1430
C		XMAX(2) = 100.00	GLIP1440
C		RETURN	GLIP1450
C			GLIP1460
C	16	GOLDSTEIN-PRICE FUNCTION (N = 2)	GLIP1470
C			GLIP1480
C	161	CONTINUE	GLIP1490
C		N = 2	GLIP1500
C		V0 = 1.00	GLIP1510
C		VMIN = -2.00	GLIP1520
C		VMAX = 2.00	GLIP1530
C		GO TO 100	GLIP1540
C			GLIP1550
C	17	PENALIZED BRANIN FUNCTION (N = 2)	GLIP1560
C			GLIP1570
C	171	CONTINUE	GLIP1580
C		N = 2	GLIP1590
C		X0(1) = 2.000	GLIP1600
C		X0(2) = 7.500	GLIP1610
C		XMIN(1) = -5.00	GLIP1620
C		XMAX(1) = 10.00	GLIP1630
C		XMIN(2) = 0.00	GLIP1640
C		XMAX(2) = 15.00	GLIP1650
C		RETURN	GLIP1660
C			GLIP1670
C	18	PENALIZED SHEKEL FUNCTION, N = 5 (N = 4)	GLIP1680
C			GLIP1690
C	181	CONTINUE	GLIP1700
C			GLIP1710

30	CONTINUE	GLIP0580
	N = 1	GLIP0590
	VO = 0.00	GLIP0600
	VMIN = -10.00	GLIP0610
	VMAX = 10.00	GLIP0620
	GO TO 100	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) = 1.00	GLIP0690
	X0(2) = 0.00	GLIP0700
	VMIN = -10.00	GLIP0710
	VMAX = 10.00	GLIP0720
	GO TO 700	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) = -3.00	GLIP0790
	X0(2) = 0.00	GLIP0800
	XMIN(1) = -15.00	GLIP0810
	XMAX(1) = 25.00	GLIP0820
	XMIN(2) = -5.00	GLIP0830
	XMAX(2) = 15.00	GLIP0840
	RETURN	GLIP0850
C		GLIP0860
C	6 SIX-HUMP CAMEL FUNCTION (N = 2)	GLIP0870
C		GLIP0880
60	CONTINUE	GLIP0890
	N = 2	GLIP0900
	VO = 0.00	GLIP0910
	XMIN(1) = -3.00	GLIP0920
	XMAX(1) = 3.00	GLIP0930
	XMIN(2) = -2.00	GLIP0940
	XMAX(2) = 2.00	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
	VO = 0.00	GLIP1100
	VMIN = -10.00	GLIP1110
	VMAX = 10.00	GLIP1120
	GO TO 1100	GLIP1130
		GLIP1140

```

SUBROUTINE GLOMIP (NPROB, N, X0, XMIN, XMAX)
C
C THE SUBROUTINE GLOMIP PROVIDES THE CODING FOR THE NUMBER
C OF VARIABLES, THE INITIAL POINT, AND THE OBSERVATION REGION
C TO BE USED, TOGETHER WITH THE 37 TEST FUNCTIONS GIVEN BY
C SUBROUTINE GLOMIF, TO DEFINE 37 TEST PROBLEMS FOR GLOBAL
C MINIMIZATION SOFTWARE.
C
C THE SUBROUTINE GLOMIP RETURNS IN N, X0, AND XMIN,
C XMAX THE NUMBER OF VARIABLES, THE INITIAL POINT, AND THE
C BOUNDARIES OF THE OBSERVATION REGION.
C
C CALLING STATEMENT
C CALL GLOMIP (NPROB, N, X0, XMIN, XMAX)
C
C DESCRIPTION OF THE CALL PARAMETERS
C (THE FORTRAN IMPLICIT TYPE DEFINITION FOR INTEGERS IS USED.
C ALL NON-INTEGER PARAMETERS ARE DOUBLE-PRECISION).
C
C NPROB IS THE (INPUT) NUMBER OF THE TEST PROBLEM TO BE
C CONSIDERED
C N IS THE (OUTPUT) NUMBER OF VARIABLES (DIMENSION) OF
C THE PROBLEM
C XMIN, XMAX ARE THE (OUTPUT) N-VECTORS CONTAINING THE LEFT
C AND RIGHT BOUNDARIES OF THE OBSERVATION REGION
C DEFINED BY THE POINTS  $X = (X_1, \dots, X_N)$  SUCH THAT
C  $XMIN(I) \leq X(I) \leq XMAX(I)$ ,  $I = 1, \dots, N$ 
C
C DOUBLE PRECISION X0, XMIN, XMAX
C DOUBLE PRECISION V0, VMIN, VMAX
C
C DIMENSION X0(1), XMIN(1), XMAX(1)
C
C 60 TO (10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120, 130, 140, 150,
C 1 160, 170, 180, 190, 200, 210, 220, 230, 240, 250, 260, 270, 280,
C 2 290, 300, 310, 320, 330, 340, 350, 360, 370), NPROB
C
C 1 A FOURTH-ORDER POLYNOMIAL (N = 1)
C
C 10 CONTINUE
C N = 1
C V0 = 1.000
C VMIN = -10.00
C VMAX = 10.00
C GO TO 300
C
C 2 GOLDSTEIN SIXTH ORDER POLYNOMIAL (N = 1)
C
C 20 CONTINUE
C N = 1
C V0 = 0.00
C VMIN = -4.00
C VMAX = 4.00
C GO TO 300
C
C 3 ONE-DIMENSIONAL PENALIZED SHUBERT FUNCTION (N = 1)
C

```

```

GLIP0010
GLIP0020
GLIP0030
GLIP0040
GLIP0050
GLIP0060
GLIP0070
GLIP0080
GLIP0090
GLIP0100
GLIP0110
GLIP0120
GLIP0130
GLIP0140
GLIP0150
GLIP0160
GLIP0170
GLIP0180
GLIP0190
GLIP0200
GLIP0210
GLIP0220
GLIP0230
GLIP0240
GLIP0250
GLIP0260
GLIP0270
GLIP0280
GLIP0290
GLIP0300
GLIP0310
GLIP0320
GLIP0330
GLIP0340
GLIP0350
GLIP0360
GLIP0370
GLIP0380
GLIP0390
GLIP0400
GLIP0410
GLIP0420
GLIP0430
GLIP0440
GLIP0450
GLIP0460
GLIP0470
GLIP0480
GLIP0490
GLIP0500
GLIP0510
GLIP0520
GLIP0530
GLIP0540
GLIP0550
GLIP0560
GLIP0570

```

```

      DO 375 I = 1, N
      FUNZ = FUNZ + DBLE(FLOAT(I)) * X(I) * X(I)
375 CONTINUE
      FUNZ = USQRT(DSQRT(FUNZ))
      RETURN
C
C 35 A FUNCTION WITH A SMALL-ATTRACTION-REGION GLOBAL MINIMUM (N = 2)
C
350 CONTINUE
      RG = P36A
      RP = P36B
      H = P36C
      PUND = P36D
      GOTO 373
C
C 37 A FUNCTION WITH A SMALL-ATTRACTION-REGION GLOBAL MINIMUM (N = 5)
C
370 CONTINUE
      RG = P37A
      RP = P37B
      H = P37C
      PUND = P37D
373 CONTINUE
      S = 7.00
      DO 377 I = 2, N
      S = S + X(I) * X(I)
377 CONTINUE
      FUNZ = S + X(1) * X(1)
      S = S + (X(1) - RG) ** 2
      IF (C.LI.RP * RP * PUND) FUNZ = FUNZ - (RG * RG + H) * EXP(-S / (RP * RP - S))
      RETURN
C
      END

```

```

GLTF4000
GLTF4010
GLTF4020
GLTF4030
GLTF4040
GLTF4050
GLTF4060
GLTF4070
GLTF4080
GLTF4090
GLTF4100
GLTF4110
GLTF4120
GLTF4130
GLTF4140
GLTF4150
GLTF4160
GLTF4170
GLTF4180
GLTF4190
GLTF4200
GLTF4210
GLTF4220
GLTF4230
GLTF4240
GLTF4250
GLTF4260
GLTF4270
GLTF4280
GLTF4290
GLTF4300
GLTF4310
GLTF4320

```

DO 205 I = 1,N	GLTF3430
Y(I) = X(I)	GLTF3440
207 CONTINUE	GLTF3450
C	GLTF3460
209 CONTINUE	GLTF3470
FUNZ = 10.00*DSIN(PI*Y(I))**2+(Y(N)-1.00)**2	GLTF3480
DO 207 I=1,N	GLTF3490
FUNZ = FUNZ+(Y(I-1)-1.00)**2*(1.00+10.00*DSIN(PI*Y(I))**2)	GLTF3500
207 CONTINUE	GLTF3510
FUNZ = FUNZ*PI/DBLE(FLOAT(N))	GLTF3520
RANGE = 10.00	GLTF3530
GO TO 347	GLTF3540
C	GLTF3550
C 29 PENALIZED LEVY-MONTALVO FUNCTION, TYPE 3, RANGE = 10 (N = 2)	GLTF3560
C	GLTF3570
290 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 = 10.00	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 = 5.00	GLTF3810
342 CONTINUE	GLTF3820
FUNZ = DSIN(7.00*PI*X(1))**2	GLTF3830
1+(X(N)-1.00)**2*(1.00+DSIN(2.00*PI*X(N))**2)	GLTF3840
DO 340 I=1,N	GLTF3850
FUNZ = FUNZ+(X(I-1)-1.00)**2*(1.00+DSIN(3.00*PI*X(I))**2)	GLTF3860
345 CONTINUE	GLTF3870
FUNZ = FUNZ*0.100	GLTF3880
347 CONTINUE	GLTF3890
DO 347 I = 1,N	GLTF3900
IF (ABS(X(I))>RANGE)	GLTF3910
1 FUNZ = FUNZ*PENFUN(X(I),RANGE,100.00,4)	GLTF3920
347 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 = 0.00	GLTF3990

M = 4	GLTF2860
FUNZ = 0.00	GLTF2870
DO 217 I = 1,M	GLTF2880
S = 0.00	GLTF2890
DO 213 J = 1,N	GLTF2900
S = S-P21A(I,J)*(A(J)-P21B(I,J))*2	GLTF2910
213 CONTINUE	GLTF2920
IF (S.GE.P21C) 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 = 0.00	GLTF3010
DO 223 I = 1,M	GLTF3020
S = 0.00	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,M	GLTF3100
IF (DABS(X(I)-0.500).GT.0.500)	GLTF3110
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) = 1.00+0.2500*(A(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)
C
160 CONTINUE
U = X(1)+X(2)+1.00
V = 2.00*X(1)-3.00*X(2)
UU = U*U
VV = V*V
FUN4 = (1.00+UU)*(36.00-20.00*U+3.00*UU)
1 FUN5 = (37.00+VV)*(18.00-16.00*V+3.00*VV)
RETURN
C
C 17 PENALIZED BRANIN FUNCTION (N = 2)
C
170 CONTINUE
FUNZ = (X(2)-P17A*(X(1)/P1)**2+(5.00/P1)*X(1)-6.00)**2
1 +10.00*(1.00-1.00/(8.00+P1))+DCOS(X(1))+10.00
IF (DABS(X(1)-2.500).GT.7.500)
1 FUNZ = FUNZ+PENFUN(X(1)-2.500,7.500,100.00,2)
IF (DABS(X(2)-7.500).GT.7.500)
1 FUNZ = FUNZ+PENFUN(X(2)-7.500,7.500,100.00,2)
RETURN
C
C 18 PENALIZED SHEKEL FUNCTION, M = 5 (N = 4)
C
180 CONTINUE
M = 5
60 T0 203
C
C 19 PENALIZED SHEKEL FUNCTION, M = 7 (N = 4)
C
190 CONTINUE
M = 7
60 T0 203
C
C 20 PENALIZED SHEKEL FUNCTION, M = 10 (N = 4)
C
200 CONTINUE
M = 10
C
203 CONTINUE
FUNZ = 0.00
DO 207 I = 1,M
S = P203(I)
DO 205 J = 1,M
S = S+(X(J)-P20A(I,J))**2
205 CONTINUE
FUNZ = FUNZ + 1.00/S
207 CONTINUE
DO 209 I = 1,M
IF (DABS(X(I)-5.00).GT.5.00)
1 FUNZ = FUNZ+PENFUN(X(I)-5.00,5.00,100.00,2)
209 CONTINUE
RETURN
C
C 21 PENALIZED THREE-DIMENSIONAL HARTMAN FUNCTION (N = 3)
C
210 CONTINUE
GLTF2290
GLTF2300
GLTF2310
GLTF2320
GLTF2330
GLTF2340
GLTF2350
GLTF2360
GLTF2370
GLTF2380
GLTF2390
GLTF2400
GLTF2410
GLTF2420
GLTF2430
GLTF2440
GLTF2450
GLTF2460
GLTF2470
GLTF2480
GLTF2490
GLTF2500
GLTF2510
GLTF2520
GLTF2530
GLTF2540
GLTF2550
GLTF2560
GLTF2570
GLTF2580
GLTF2590
GLTF2600
GLTF2610
GLTF2620
GLTF2630
GLTF2640
GLTF2650
GLTF2660
GLTF2670
GLTF2680
GLTF2690
GLTF2700
GLTF2710
GLTF2720
GLTF2730
GLTF2740
GLTF2750
GLTF2760
GLTF2770
GLTF2780
GLTF2790
GLTF2800
GLTF2810
GLTF2820
GLTF2830
GLTF2840
GLTF2850

```

```

          S2 = S2+D1*DCOS((D1+1.40)*A(2)+D1)
95 CONTINUE
      FUNZ = FUNZ+S1*S2
      IF (DABS(X(1)).GT.10.00) FUNZ = FUNZ+PENFUN(X(1),10.00,100.00,2)
      IF (DABS(X(2)).GT.10.00) FUNZ = FUNZ+PENFUN(X(2),10.00,100.00,2)
      RETURN
C
C 10 A FUNCTION WITH THREE ILL-CONDITIONED MINIMA, A=10 (N = 2)
C
C 100 CONTINUE
      A = 1.01
      B = 1.0-1
      GO TO 105
C
C 11 A FUNCTION WITH THREE ILL-CONDITIONED MINIMA, A=10**2 (N = 2)
C
C 110 CONTINUE
      A = 1.02
      B = 1.0-2
      GO TO 155
C
C 12 A FUNCTION WITH THREE ILL-CONDITIONED MINIMA, A=10**3 (N = 2)
C
C 120 CONTINUE
      A = 1.03
      B = 1.0-3
      GO TO 155
C
C 13 A FUNCTION WITH THREE ILL-CONDITIONED MINIMA, A=10**4 (N = 2)
C
C 130 CONTINUE
      A = 1.04
      B = 1.0-4
      GO TO 155
C
C 14 A FUNCTION WITH THREE ILL-CONDITIONED MINIMA, A=10**5 (N = 2)
C
C 140 CONTINUE
      A = 1.05
      B = 1.0-5
      GO TO 155
C
C 15 A FUNCTION WITH THREE ILL-CONDITIONED MINIMA, A=10**6 (N = 2)
C
C 150 CONTINUE
      A = 1.06
      B = 1.0-6
C
C 155 CONTINUE
      XX = X(1)*X(1)
      YY = X(2)*X(2)
      R2 = XX+YY
      R4 = R2*R2
      R6 = R4*R4
      FUNZ = A*XX+YY-R4+R6
      RETURN
C

```

GLTF1720
 GLTF1730
 GLTF1740
 GLTF1750
 GLTF1760
 GLTF1770
 GLTF1780
 GLTF1790
 GLTF1800
 GLTF1810
 GLTF1820
 GLTF1830
 GLTF1840
 GLTF1850
 GLTF1860
 GLTF1870
 GLTF1880
 GLTF1890
 GLTF1900
 GLTF1910
 GLTF1920
 GLTF1930
 GLTF1940
 GLTF1950
 GLTF1960
 GLTF1970
 GLTF1980
 GLTF1990
 GLTF2000
 GLTF2010
 GLTF2020
 GLTF2030
 GLTF2040
 GLTF2050
 GLTF2060
 GLTF2070
 GLTF2080
 GLTF2090
 GLTF2100
 GLTF2110
 GLTF2120
 GLTF2130
 GLTF2140
 GLTF2150
 GLTF2160
 GLTF2170
 GLTF2180
 GLTF2190
 GLTF2200
 GLTF2210
 GLTF2220
 GLTF2230
 GLTF2240
 GLTF2250
 GLTF2260
 GLTF2270
 GLTF2280

```

C
30 CONTINUE
FUNZ = 0.00
DO 35 I = 1,5
  D1 = DBLE(FLCAT(1))
  FUNZ = FUNZ+D1*DCOS((D1+1.00)*X(1)+D1)
35 CONTINUE
IF (DABS(X(1)) .GT. 10.00) FUNZ = FUNZ+PENFUN(X(1),10.00,100.00,2)
RETURN
GLTF1150
GLTF1160
GLTF1170
GLTF1180
GLTF1190
GLTF1200
GLTF1210
GLTF1220
GLTF1230
C
4 A FOURTH ORDER POLYNOMIAL IN TWO VARIABLES (N = 2)
GLTF1240
GLTF1250
GLTF1260
C
47 CONTINUE
XX = X(1)*X(1)
YY = X(2)*X(2)
FUNZ = 0.2500*XX*XX-0.5000*XA+P4A*X(1)+0.5000*YY
RETURN
GLTF1270
GLTF1280
GLTF1290
GLTF1300
GLTF1310
GLTF1320
C
5 A FUNCTION WITH A SINGLE ROW OF LOCAL MINIMA (N = 2)
GLTF1330
GLTF1340
GLTF1350
C
50 CONTINUE
FUNZ = 0.5000*(P5A*X(1)+A(1)+1.00-DCOS(2.00*X(1)))+X(2)*X(2)
RETURN
GLTF1360
GLTF1370
GLTF1380
C
6 SIX-HUMP CAMEL FUNCTION (N = 2)
GLTF1390
GLTF1400
GLTF1410
C
60 CONTINUE
XX = X(1)*X(1)
YY = X(2)*X(2)
FUNZ = ((XX/7.00)-(2.00+P5A))*XX+4.00)*XX+X(1)*X(2)
1 +4.00*(YY-1.00)*YY
RETURN
GLTF1420
GLTF1430
GLTF1440
GLTF1450
GLTF1460
GLTF1470
C
7 TWO-DIMENSIONAL PENALIZED SHUBERT FUNCTION, BETA = 0 (N = 2)
GLTF1480
GLTF1490
GLTF1500
C
70 CONTINUE
BETA = 0.00
GO TO 93
GLTF1510
GLTF1520
GLTF1530
C
8 TWO-DIMENSIONAL PENALIZED SHUBERT FUNCTION, BETA = 1/2 (N = 2)
GLTF1540
GLTF1550
GLTF1560
C
80 CONTINUE
BETA = 0.500
GO TO 93
GLTF1570
GLTF1580
GLTF1590
C
9 TWO-DIMENSIONAL PENALIZED SHUBERT FUNCTION, BETA = 1 (N = 2)
GLTF1600
GLTF1610
GLTF1620
C
97 CONTINUE
BETA = 1.000
GLTF1630
GLTF1640
C
97 CONTINUE
FUNZ = ((X(1)-P5A)**2+(X(2)-P5B)**2)*BETA
11 = 0.00
12 = 0.00
DO 99 I = 1,5
  D1 = DBLE(FLCAT(1))
  D1 = D1+D1*DCOS((D1+1.00)*X(1)+D1)
GLTF1650
GLTF1660
GLTF1670
GLTF1680
GLTF1690
GLTF1700
GLTF1710

```



```

DATA P1 /3.1415,265356979628460/
DATA P1A /0.100/
DATA P4A /0.100/
DATA P5A /0.100/
DATA P6A /0.100/
DATA P9A,P9B /-1.425128426319760970800,-0.8003211004719731246600/
DATA P17A /1.27500/
DATA P20A /4.00,1.00,6.00,6.00,3.00,2.00,5.00,8.00,6.00,7.00,
1 4.00,1.00,6.00,6.00,7.00,9.00,5.00,1.00,2.00,3.00,
2 4.00,1.00,6.00,6.00,3.00,2.00,3.00,8.00,6.00,7.00,
3 4.00,1.00,8.00,6.00,7.00,9.00,3.00,1.00,2.00,3.00/
DATA P20B /0.100,0.200,0.200,0.400,0.400,0.600,0.300,0.700,
1 0.500,0.500/
DATA P21A /3.00,0.100,3.00,0.100,
1 1.00,1.00,1.00,1.00,
2 30.00,35.00,30.00,35.00/
DATA P219 /0.363200,0.409900,0.109100,0.0381500,
1 0.117000,0.407700,0.873200,0.574300,
2 0.267300,0.747000,0.554700,0.882800/
DATA P21C /1.00,1.200,3.00,3.200/
DATA P21D /-69.00/
DATA P22A /10.00,0.05,0.3.00,17.00,3.00,10.00,3.50,8.00,
1 17.00,17.00,1.700,0.0500,3.500,0.100,10.00,10.00,
2 1.700,6.00,1.700,0.100,8.00,14.00,8.00,14.00/
DATA P22B /0.131200,0.232900,0.234800,0.404700,
1 0.169600,0.413500,0.145100,0.882300,
2 0.556900,0.530700,0.352200,0.873200,
3 0.012400,0.373000,0.288300,0.574300,
4 0.523100,0.100400,0.304700,0.109100,
5 0.535500,0.979100,0.665000,0.038100/
DATA P22C /1.00,1.200,3.00,3.200/
DATA P22D /-69.00/
DATA P30A,P30B,P30C,P30D /10.00,1.00,10.00,0.9800/
DATA P37A,P37B,P37C,P37D /10.00,1.00,10.00,0.9800/
C
C PENALIZATION FUNCTION
C
C PENFUN (VAR,RAN,FACT,IEAP) = FACT*(DABS(VAR)-RAN)**IEAP
C
C
C GO TO (10,20,30,40,50,60,70,80,90,100,110,120,130,140,150,
1 160,170,180,190,200,210,220,230,240,250,260,270,280,
2 290,300,310,320,330,340,350,360,370), NPROB
C
C 1 A FOURTH ORDER POLYNOMIAL (N = 1)
C
C 10 CONTINUE
FUN1 = ((C.2500*X(1)+X(1)-0.500)*X(1)+P1A)*X(1)
RETURN
C
C 2 GOLDSTEIN SIXTH ORDER POLYNOMIAL (N = 1)
C
C 20 CONTINUE
YX = X(1)*X(1)
FUN2 = ((AX-10.00)*YX+27.00)*AX+250.00
RETURN
C
C 3 ONE-DIMENSIONAL PENALIZED CHI-SQUARED FUNCTION (N = 1)
C

```

```

GLTF0580
GLTF0590
GLTF0600
GLTF0610
GLTF0620
GLTF0630
GLTF0640
GLTF0650
GLTF0660
GLTF0670
GLTF0680
GLTF0690
GLTF0700
GLTF0710
GLTF0720
GLTF0730
GLTF0740
GLTF0750
GLTF0760
GLTF0770
GLTF0780
GLTF0790
GLTF0800
GLTF0810
GLTF0820
GLTF0830
GLTF0840
GLTF0850
GLTF0860
GLTF0870
GLTF0880
GLTF0890
GLTF0900
GLTF0910
GLTF0920
GLTF0930
GLTF0940
GLTF0950
GLTF0960
GLTF0970
GLTF0980
GLTF0990
GLTF1000
GLTF1010
GLTF1020
GLTF1030
GLTF1040
GLTF1050
GLTF1060
GLTF1070
GLTF1080
GLTF1090
GLTF1100
GLTF1110
GLTF1120
GLTF1130
GLTF1140

```

```

SUBROUTINE GLOMTF (NPROB, N, X, FUNZ)
C
C THE SUBROUTINE GLOMTF PROVIDES THE CODING OF 37 REAL-VALUED
C FUNCTIONS OF N REAL VARIABLES, TO BE USED, TOGETHER WITH THE
C SUBROUTINE GLOMIP, TO DEFINE 37 TEST PROBLEMS FOR GLOBAL
C MINIMIZATION SOFTWARE.
C
C THE SUBROUTINE GLOMTF RETURNS IN FUNZ THE FUNCTION VALUE
C AT THE POINT X = (X1,X2,...,XN) FOR THE FUNCTION DEFINED BY
C PROBLEM NUMBER NPROB.
C
C CALLING STATEMENT
C
C CALL GLOMTF (NPROB, N, X, FUNZ)
C
C DESCRIPTION OF THE CALL PARAMETERS
C (THE FORTRAN IMPLICIT TYPE DEFINITION FOR INTEGERS IS USED.
C ALL NON-INTEGER PARAMETERS ARE DOUBLE-PRECISION).
C
C NPROB IS THE (INPUT) TEST-PROBLEM NUMBER
C
C N IS THE (INPUT) DIMENSION OF THE PROBLEM
C
C X IS AN (INPUT) N-VECTOR CONTAINING THE INDEPENDENT VARIABLES
C
C FUNZ IS THE (OUTPUT) VALUE AT X OF THE FUNCTION DEFINED BY
C PROBLEM NUMBER NPROB.
C
C
C DOUBLE PRECISION X
C DOUBLE PRECISION FUNZ
C
C DOUBLE PRECISION PI
C DOUBLE PRECISION P1A
C DOUBLE PRECISION P4A
C DOUBLE PRECISION P5A
C DOUBLE PRECISION P6A
C DOUBLE PRECISION P9A,P9B
C DOUBLE PRECISION P17A
C DOUBLE PRECISION P20A,P20B
C DOUBLE PRECISION P21A,P21B,P21C,P21D
C DOUBLE PRECISION P22A,P22B,P22C,P22D
C DOUBLE PRECISION P36A,P36B,P36C,P36D
C DOUBLE PRECISION P37A,P37B,P37C,P37D
C DOUBLE PRECISION Y
C DOUBLE PRECISION X1,D1,YY,BLTA,S1,S2,A,D,R2,R4,R8,U,V,UU,VV,S
C DOUBLE PRECISION RG,RP,H,PUND,RANGE
C DOUBLE PRECISION FACT,VAR,P,N,FUN,RAN
C
C DIMENSION X(N)
C
C DIMENSION P1(2A(10,4),P20(10)
C DIMENSION P11A(4,2),P11B(4,2),P21C(4)
C DIMENSION P12A(4,2),P12B(4,2),P22C(4)
C DIMENSION Y(10)
C
GLTF0010
GLTF0020
GLTF0030
GLTF0040
GLTF0050
GLTF0060
GLTF0070
GLTF0080
GLTF0090
GLTF0100
GLTF0110
GLTF0120
GLTF0130
GLTF0140
GLTF0150
GLTF0160
GLTF0170
GLTF0180
GLTF0190
GLTF0200
GLTF0210
GLTF0220
GLTF0230
GLTF0240
GLTF0250
GLTF0260
GLTF0270
GLTF0280
GLTF0290
GLTF0300
GLTF0310
GLTF0320
GLTF0330
GLTF0340
GLTF0350
GLTF0360
GLTF0370
GLTF0380
GLTF0390
GLTF0400
GLTF0410
GLTF0420
GLTF0430
GLTF0440
GLTF0450
GLTF0460
GLTF0470
GLTF0480
GLTF0490
GLTF0500
GLTF0510
GLTF0520
GLTF0530
GLTF0540
GLTF0550
GLTF0560
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 Cirilli

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. Cirilli, 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 $f: \mathbb{R}^N \rightarrow \mathbb{R}$ 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

$$(2.1) \quad f(\underline{x}^*) \leq f(\underline{x}) \quad \forall \underline{x} \in \mathbb{R}^N$$

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 $\underline{g}: \mathbb{R}^N \rightarrow \mathbb{R}^N$ 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

$$(2.2) \quad d\underline{\xi} = -\nabla f(\underline{\xi})dt + \underline{\sigma}d\underline{w}$$

where ∇f is the gradient of f and $\underline{w}(t)$ is a standard N -dimensional Wiener process, $t \in \mathbb{R}$.

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 εdw , 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

$$(2.3) \quad \lim_{\|\underline{x}\|_2 \rightarrow \infty} f(\underline{x}) = +\infty$$

in such a way that:

$$(2.4) \quad \int_{\mathbb{R}^N} e^{-\alpha^2 f(\underline{x})} d\underline{x} < \infty \quad \forall \alpha \in (\mathbb{R} \setminus \{0\})$$

and that the minimizers of f are isolated and non degenerate.

It is well known that if $\underline{x}^\varepsilon(t)$ 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 $\underline{x}^\varepsilon(t)$ approaches as $t \rightarrow \infty$ the limit density $p_\sigma^\varepsilon(\underline{x})$ where

$$(2.5) \quad p_\sigma^\varepsilon(\underline{x}) = A_\sigma e^{-\frac{2}{\varepsilon^2} f(\underline{x})}$$

where A_σ is a normalization constant. The way in which $p^\varepsilon(t, \underline{x})$ for a class of one-dimensional systems approaches $p_\sigma^\varepsilon(\underline{x})$ has been studied in detail by considering the spectrum of the corresponding Fokker-Planck operators in [8].

We note that p_t^x is independent of x_0 and that as $\epsilon \rightarrow 0$ p_t^x becomes more concentrated at the global minimizers of f . That is,

$$(2.6) \quad \lim_{t \rightarrow \infty} \mathbb{P}(\underline{X}(t) = \underline{X}_*) \quad \text{in law}$$

where \underline{X}_* has a probability density given by (2.5) and

$$(2.7) \quad \lim_{\epsilon \rightarrow 0} \mathbb{P}_\epsilon^x = \mathbb{P}_*^x \quad \text{in law}$$

where \mathbb{P}_*^x 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 $\frac{d^2 f}{dx^2}(x_1) = c_1 > 0$, $i = 1, 2$, we have (in distribution sense)

$$(2.8) \quad \lim_{\epsilon \rightarrow 0} p_\epsilon^x(x) = \frac{c_2}{c_1 + c_2} \delta(x - x_1) + \frac{c_1}{c_1 + c_2} \delta(x - x_2)$$

where $\delta = (1 + \sqrt{c_1/c_2})^{-1}$. In order to obtain the global minimizers of f as asymptotic values as $t \rightarrow \infty$ of a sample trajectory of a suitable system of stochastic differential equations it seems natural to try to perform the limit $t \rightarrow \infty$ (i.e. (2.6)) and the limit $\epsilon \rightarrow 0$ (i.e. (2.7)) together.

That is, we want to consider:

$$(2.9) \quad d\underline{X} = -\nabla f(\underline{X})dt + \epsilon(t)d\underline{w}$$

with initial condition

$$(2.10) \quad \underline{X}(0) = \underline{x}_0$$

where

$$(2.11) \quad \lim_{t \rightarrow \infty} \epsilon(t) = 0.$$

In physical terms condition (2.11) means that the temperature T is decreased to 0 (absolute zero) when $t \rightarrow \infty$, 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 $\beta(t)$ must go to zero, in order to have that when $t \rightarrow \infty$, the solution $\underline{x}(t)$ 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 $\underline{x}(t)$ to a random variable concentrated at the global minimizers of f , and the convergence with probability one of the paths of $\underline{x}(t)$ 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 $\underline{x}(t_k)$ is approximated by the solution \underline{x}_k of the following finite difference equations:

$$(2.12) \quad \underline{x}_{k+1} - \underline{x}_k = -h_k \nabla f(\underline{x}_k) + \epsilon(t_k)(w_{k+1} - w_k) \quad k = 0, 1, 2, \dots$$

$$(2.13) \quad \underline{x}_0 = \underline{x}_j$$

where $t_0 = 0$, $t_k = \sum_{i=0}^{k-1} h_i$, $h_k > 0$, and $w_k = w(t_k)$, $k = 0, 1, 2, \dots$.

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 $\underline{x}(t)$ should go to zero very slowly when $t \rightarrow \infty$, 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 $\epsilon(t_k)(w_{k+1} - w_k)$ to the deterministic term $-h_k \nabla f(\underline{x}_k)$, 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 $\nabla f(\underline{x}_k)$ is frequently lost.

In order to avoid this inconvenience we substitute the gradient $\nabla f(\underline{x})$ with a "random gradient" as follows. Let \underline{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 $\underline{v} \in \mathbb{R}^N$ its projection along \underline{r} is such that:

$$(2.14) \quad N \cdot E(\langle \underline{r}, \underline{v} \rangle \underline{r}) = \underline{v}$$

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 $\nabla f(\underline{x}_k)$ with the "random gradient"

$$(2.15) \quad \underline{z}(\underline{z}_k) = N < \underline{r}, \quad \forall f(\underline{z}_k) > \underline{r}.$$

We note that since $\frac{1}{N} \underline{z}(\underline{z}_k)$ 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 $\underline{z}(\underline{z})$).

Therefore, the paths are computed approximating $\underline{z}(t_k)$ with the solution \underline{z}_k of the following differences equations:

$$(2.16) \quad \underline{z}_{k+1} - \underline{z}_k = -h_k \underline{z}(\underline{z}_k) + \psi(t_k)(w_{k+1} - w_k) \quad k = 0, 1, 2, \dots$$

$$(2.17) \quad \underline{z}_0 = \underline{x}_0$$

where $\underline{z}(\underline{z}_k)$ is a finite difference (forward or central) approximation to $\underline{z}(\underline{z}_k)$.

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 $\epsilon(0)$ (note that even if the starting values $\epsilon(0)$ 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 σ_p , 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 σ_p and Δx_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

$$(3.2.1.1) \quad \underline{x}_{k+1} = \underline{x}_k - h_k \underline{f}(\underline{x}_k) + \varepsilon_p \sqrt{h_k} \underline{u}_k \quad (k = 0, 1, 2, \dots)$$

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 $\underline{f}(\underline{x}_k)$ is described in the next section.

The basic step (3.2.1.1) is actually performed in two half-steps

$$(3.2.1.2) \quad \bar{z}'_k = \bar{z}_k - h_k \bar{z}(\bar{z}_k) \quad (\text{first half-step})$$

and

$$(3.2.1.3) \quad \bar{z}_{k+1} = \bar{z}'_k + \tau_p \sqrt{h_k} u_k \quad (\text{second half-step})$$

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 $\bar{z}(\bar{z}_k)$.

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

$$(3.2.2.1) \quad \begin{cases} \underline{F}_k = f(\bar{z}_k + \underline{s}_k) - f(\bar{z}_k) \\ \underline{G}_k = \frac{1}{2} [f(\bar{z}_k + \underline{s}_k) - f(\bar{z}_k - \underline{s}_k)] \end{cases}$$

the forward- and central-differences directional derivatives

$$(3.2.2.2) \quad \underline{F}_k = \underline{F}_k / \Delta x_k \quad \underline{G}_k = \underline{G}_k / \Delta x_k$$

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. 4.3.

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

$$|x-y| \leq \tau_{\text{REL}} (|x| + |y|)/2$$

b) Absolute difference criterion

$$|x-y| \leq \tau_{\text{ABS}}$$

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 $N \times N$ 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{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 $f(\underline{\tilde{x}}) = f(\underline{x}) = f(A\underline{x} + \underline{b})$, 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 $\underline{\tilde{x}}$).

The updating of A is obtained by means of an updating matrix E_A , and is 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 $\lceil N \rceil$.

The updating matrix E_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 $\underline{g}_i(i)$, $i = 1, 2, \dots, N_g$, be the column vectors of the components of all the N_g finite-difference random gradients \underline{g} (\underline{g}^F or \underline{g}^G) 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

$$\bar{g} = \frac{1}{N_g} \sum_{i=1}^{N_g} \underline{g}_i(i) \quad (1)$$

and the estimated covariance matrix

$$C = \frac{1}{N_g} \sum_{i=1}^{N_g} (\underline{g}_i(i) - \bar{g})(\underline{g}_i(i) - \bar{g})^T$$

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 derivative of f is, 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

$$R_1^{\text{MIN}} \leq x_1 \leq R_1^{\text{MAX}}, \quad i = 1, 2, \dots, N,$$

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:

$$\alpha = 10^{-t} \text{ if } t \leq \lceil (2/5) N_{\text{TRIAL}} \rceil$$

$$\alpha = 10^{-5} \text{ otherwise,}$$

where $\lceil [x] \rceil$ is the smallest integer not smaller than x

unsuccessful non-uniform stop: $\alpha = 10^{-5}$

The initial point \underline{x}_0 is selected as follows:

if $t \leq \lceil (2/5) N_{\text{TRIAL}} \rceil$ take the value of \underline{x}_0 at algorithm start

otherwise take $\underline{x}_0 = \underline{x}_{\text{OPT}}$

where $\underline{x}_{\text{OPT}}$ 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 Δh 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 value f_{OPT} found so far from algorithm start: if f_{TMIN} 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 ε_p for the second continuation of a branched trajectory
- value of ε_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 stop (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 ε_p for the second continuation of a branched trajectory was described in sect. 3.2.7.

The value of ε_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_p for σ_p is as follows:

for the first trial and for any trial following an unsuccessful trial

$F_p = 10^{x-1/2}$ where x is a random sample from a standard normal distribution

for all other trials

$F_p = 2^{y-1/2}$ 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 Δx_k is:

$F_{\Delta x} = 10^{3z}$ where z is a random sample from a standard normal distribution.

5.2.8 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_{TMAX} , and the minimum, f_{TMIN} , among the trial final values satisfy at least one of the criteria in sect. 5.2.15, the relative difference criterion with a given stopping tolerance τ_{REL} and/or the absolute difference criterion with given stopping tolerance τ_{ABS} ("uniform stop at the level f_{TMIN} ").

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_{TMIN}

From the point of view of the noise coefficient σ_p a trajectory with larger σ_p is considered better if the comparison is made in an early observation period (as long as $k_p \leq M_p \cdot I_b$, 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_{p0} , and then every M_p periods (k_{p0} and M_p are given integers); i.e. exceptional observation periods are those numbered

$$k_p = k_{p0} + M_p \cdot j \quad (j = 0, 1, 2, \dots)$$

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 σ_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: $\epsilon = 0.1$

We finally remark that h_k and Δx_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) $N_{hp} = 1 + \lceil \log_2(k_p) \rceil$ ("short" duration)
- 2) $N_{hp} = \lceil k_p^{1/2} \rceil$ ("medium-size" duration)
- 3) $N_{hp} = k_p$ ("long" duration)

where $k_p = 1, 2, \dots$, and $\lceil x \rceil$ 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

$$\tau_{R1} = 10^{-11} \quad \text{and} \quad \tau_{R2} = 10^{-5}, \quad \text{and take}$$

$$z = 2 \quad \text{if} \quad f_k \quad \text{and} \quad \hat{f}_k \quad \text{are "equal" within} \quad \tau_{R1}$$

$$z = \frac{1}{2} \quad \text{if} \quad f_k \quad \text{and} \quad \hat{f}_k \quad \text{are not "equal" within} \quad \tau_{R2}$$

$$z = 1 \quad \text{otherwise.}$$

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:

$$\gamma = 1/1.05 \quad \text{for the first attempt to the first half-step}$$

$$\gamma = \frac{1}{2} \quad \text{for the second attempt}$$

$$\gamma = 1/10 \quad \text{for all other attempts}$$

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 $r > 0$

$$\gamma = 1 \quad (\text{if } a \leq 2r)$$

$$\gamma = 1.1 \quad (\text{if } 2r < a \leq 3r)$$

$$\gamma = 2 \quad (\text{if } 3r < a)$$

If $r = 0$

$$\gamma = 2 \quad (\text{if } a = 1)$$

$$\gamma = 10 \quad (\text{if } a > 1)$$

- 0a. If the half-step is rejected: reject also the first half-step, update (decrease) h_k , and go back to 1.
- 0b. 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_k becomes smaller than 10^{-30})).

5.2.4 The updating of h_k and Δx_k .

The time-integration steplength h_k and the spatial discretization increment Δx_k 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. 5.2.3, is as follows:

Updating factors γ for Δx_k

In phase 1b: $\gamma = 10^2$

In phase 2a: $\gamma = 10$

In phase 1b: $\gamma = 10^{-7}$

In phase 3 the value of γ depends on the magnitude of the current estimated function increment $\Delta f_k = \tau_k \Delta x_k$ (where τ_k is τ_k^F or τ_k^C as appropriate), and the function value $f_k = f(\tau_k)$.

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. 5.2.2).
 - 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{r}_k^F (eq. (5.2.2.2)).
 - 2a. If the computed value of $(\underline{r}_k^F)^2$ is zero (due to the finite arithmetics): update (increase) Δx_k and go back to 1a.
3. Compute the first half-step with \underline{r}_k^F .
Compute $\Delta' f_k$ (eq. (5.2.3.1)).
 - 3a. If $|\Delta' f_k| \leq |\underline{r}_k^F| \Delta x_k$
accept the first half-step and jump to 5.
4. Compute the first half-step with \underline{r}_k^C to check the appropriateness of Δx_k .
Compute $\Delta'' f_k$ (eq. (5.2.3.1)).
 - 4a. If $|\Delta'' f_k| > |\underline{r}_k^F| - |\underline{r}_k^C| \Delta x_k$
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. (5.2.3.2)).

and the forward- and central-differences random gradients

$$(3.2.2.5) \quad \underline{\Delta}_k^F = N \underline{r}_k^F \underline{r}_k \quad \underline{\Delta}_k^C = N \underline{r}_k^C \underline{r}_k$$

We use $\underline{\Delta}_k^F$ or $\underline{\Delta}_k^C$ for $\underline{\Delta}(\underline{z}_k)$ 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 ($\underline{\Delta}_k^F$ or $\underline{\Delta}_k^C$ eq. (3.2.2.5)) as described below.

In either case the half-step is accepted or rejected according to the function increment

$$(3.2.3.1) \quad \Delta'f_k = f(\underline{z}_k') - f(\underline{z}_k)$$

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

$$(3.2.3.2) \quad \Delta''f_k = f(\underline{z}_{k+1}') - f(\underline{z}_k')$$

is positive and too large (greater than $100 \frac{\epsilon}{p}$ 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 GLOMIF 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 IOUF 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 IOUF > 0) or a failure (IOUF ≤ 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 occurs 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

NPROB	N	1		2		3		4		5	
		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	45,505	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	1,092	0	260,616	0	278,111	0	327,392	0
21	3	3,416	0	1,520	0	2,114	0	2,114	0	50,114	0
22	3	4,729	0	1,468	0	2,114	0	2,114	0	52,104	0
23	3	11,881	0	1,111	0	1,111	0	1,111	0	92,111	0

Table 1 (continued)

N _{SUB} =	INIVAC (continued)												
	1		2		3		4		5		5		
NPROB	N	Nf	Ie	Nf	Ie	Nf	Ie	Nf	Ie	Nf	Ie	Nf	Ie
24	5	8,099	0	36,057	0	47,619	0	69,901	0	92,104	0	92,104	0
25	4	11,954	0	54,212	0	71,655	0	97,724	0	191,722	0	191,722	0
26	5	43,083	0	284,104	0	347,056	0	450,102	0	464,611	0	464,611	0
27	8	2,324	0	21,124	0	75,728	0	635,990	0	654,436	0	654,436	0
28	10	50,975	0	426,171	0	454,808	0	474,323	0	479,817	0	479,817	0
29	2	25,462	0	35,675	0	98,944	0	111,447	0	167,728	0	167,728	0
30	3	15,734	0	113,789	0	177,970	0	257,904	0	286,273	0	286,273	0
31	4	11,516	0	143,757	0	208,217	0	264,834	0	296,663	0	296,663	0
32	5	50,911	0	176,840	0	275,852	0	357,089	0	679,442	0	679,442	0
33	6	53,178	0	102,652	0	272,642	0	303,267	0	454,543	0	454,543	0
34	7	14,594	0	298,256	0	357,878	0	409,949	0	520,641	0	520,641	0
35	5	33,635	0	50,348	0	70,105	0	127,091	0	183,864	0	183,864	0
36	2	3,102	0	10,176	0	23,283	0	72,931	0	79,481	0	79,481	0
37	5	6,938	0	12,469	0	25,175	0	64,639	0	92,407	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	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	8,613	0	2,511	0	1,111	0	68,127	0				
23	3	5,531	0	1,111	0	1,111	0	1,111	0	1,111	0				

Table 1 (continued)

NPROB	N	1				2				3				4				5			
		Nf	Ie	Nf	Ie	Nf	Ie	Nf	Ie	Nf	Ie	Nf	Ie	Nf	Ie	Nf	Ie	Nf	Ie		
24	3	10,050	0	18,170	0	80,567	0	109,726	0	190,920	0	227,682	0	330,192	0	474,422	0	585,249	0		
25	4	10,657	0	37,655	0	56,285	0	129,598	0	227,682	0	474,422	0	825,175	0	1,171,995	0	1,719,995	0		
26	5	59,689	0	369,912	0	460,779	0	1,171,995	0	2,019,665	0	3,167,020	0	4,744,220	0	6,855,175	0	9,719,995	0		
27	8	168,933	0	259,950	0	302,092	0	510,718	0	825,175	0	1,171,995	0	1,719,995	0	2,425,535	0	3,397,066	0		
28	10	43,466	0	393,550	0	411,854	0	685,927	0	1,048,183	0	1,518,418	0	2,199,665	0	3,066,066	0	4,171,995	0		
29	2	15,223	0	55,718	0	95,254	0	131,418	0	171,995	0	242,535	0	330,192	0	474,422	0	685,927	0		
30	3	12,641	0	54,822	0	118,927	0	201,965	0	316,702	0	474,422	0	685,927	0	1,048,183	0	1,518,418	0		
31	4	26,235	0	123,716	0	148,183	0	242,535	0	330,192	0	474,422	0	685,927	0	1,048,183	0	1,518,418	0		
32	5	35,365	0	86,758	0	186,860	0	285,371	0	397,066	0	533,175	0	719,995	0	1,048,183	0	1,518,418	0		
33	6	49,087	0	71,418	0	159,987	0	184,087	0	296,770	0	411,854	0	510,718	0	719,995	0	1,048,183	0		
34	7	50,237	0	132,768	0	173,955	0	204,081	0	348,809	0	474,422	0	685,927	0	1,048,183	0	1,518,418	0		
35	5	14,815	0	53,394	0	79,235	0	129,480	0	206,980	0	316,702	0	474,422	0	685,927	0	1,048,183	0		
36	2	3,744	0	9,574	0	23,355	0	55,947	0	82,518	0	118,927	0	168,933	0	259,950	0	369,912	0		
37	5	3,847	0	12,239	0	29,411	0	70,599	0	107,264	0	168,933	0	259,950	0	369,912	0	460,779	0		

NPROB = problem number given in sect. 4.1.

Ie = 0 success (IOFF > 0) (claimed by SIGMA)

= 1 failure (IOFF < 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	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	3	3	3	1	1
28	10	1	1	1	1	1

Table 2 (continued)

UNIVAC (continued)

$N_{\text{SUC}} =$		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

N _{SUC} =	UNIVAC					VAX				
	1	2	3	4	5	1	2	3	4	5
Totals	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 $|x_i| < 10$ $i = 1, 2, \dots, 10$.

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 DAJ837-81-C-0740

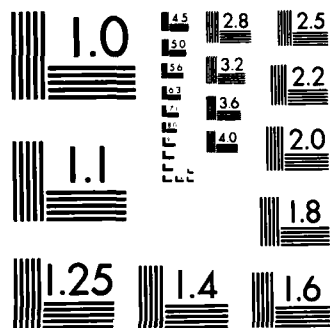
F/G 12/1

NL

END

FORM

DEC



MICROCOPY RESOLUTION TEST CHART
NBS-1963-A

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, 1985.
- [7] Schuss E.: "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] Kirkpatrick S., Gelatt C.D. Jr., Vecchi M.P.: "Optimization by simulated annealing," *Science* 220 , (1983), 671-680.
- [10] Aluffi-Pentini F., Parisi V., Zirilli F.: "Test problems for global optimization software," submitted to *A.C.M. Transactions 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, 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).

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 $f(\underline{x}) = f(x_1, \dots, x_N)$ of N real variables x_1, \dots, 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

$$\begin{aligned} \underline{\xi}_{k+1} &= \underline{\xi}_k - h_k \tilde{Y}(\underline{\xi}_k) + \varepsilon(t_k) \cdot \sqrt{h_k} \underline{u}_k, & k = 0, 1, 2, \dots \\ \underline{\xi}_0 &= \underline{x}_0 \end{aligned}$$

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

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 $\tilde{Y}(\underline{\xi}_k)$ 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 $f_{\text{TFMIN}} > f_{\text{OPT}}$ the trial is again considered unsuccessful; and if $f_{\text{TFMIN}} = f_{\text{OPT}}$ (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 σ_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 \neq -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 \neq 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 \neq 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 successful 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, X0, 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:

$$H = 10^{-10}$$

$$EPS = 1$$

$$DX = 10^{-9}$$

$$IRAND = 0$$

$$NTRAJ = 0$$

$$ISEGBR = 0$$

$$KPBR0 = 0$$

$$INKPBR = 0$$

$$NPMIN = 10$$

$$NPMAX0 = 100$$

$$INPMAX = 50$$

$$NTRIAL = \max(50, 5 \cdot NSUC)$$

$$TOLREL = 10^{-3}$$

$$TOLABS = 10^{-6}$$

$$KPASCA = 10 \quad (\text{if } N \leq 5); = 300 \quad (\text{if } N > 5)$$

$$INHP = 1$$

$$XRMIN(I) = -10^4 \quad (I = 1, \dots, N)$$

$$XRMAX(I) = 10^4 \quad (I = 1, \dots, N)$$

NPMIN (say $5 < \text{NPMIN} < 100$)

NPMAXØ (say $0 < \text{NPMAXØ} < 150$)

NTRIAL (say $\text{NTRIAL} > 50$ and $\text{NTRIAL} > 5 \cdot \text{NSUC}$)

INPMAX (say $30 < \text{INPMAX} < 100$)

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 intolerable 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 if 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 $IRAND \leq 0$ 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 ($N_{TRAJ} = 7$), and if the input value is outside the interval (3,20) N_{TRAJ} is set to the nearest extreme value).

I_{SEGBR} , $K_{PBR\emptyset}$, I_{NKPBR} 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: $I_{SEGBR} = (1+N_{TRAJ})/2$, (FORTRAN integer division), $I_{NKPBR} = 10$, $K_{PBR\emptyset} = 3$; if the input value for I_{SEGBR} is outside the interval (1, N_{TRAJ}), I_{SEGBR} is set to the nearest extreme value; and if $K_{PBR\emptyset}$ has a value not inside the interval (1, I_{NKPBR}), it is assigned the same value modulo I_{NKPBR}).

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

$N_{PMA\emptyset}$ 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 $N_{PMA\emptyset}$ of observation periods in a trial ($N_{PMA\emptyset}$).

I_{NPMAX} is the (input) increment for $N_{PMA\emptyset}$, when $N_{PMA\emptyset}$ is varied from one trial to the following one.

N_{SUC} 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}).

N_{TRIAL} 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, X \emptyset , H, EPS, DX,
 NTRAJ, ISEGBR, KPBR \emptyset , INKPBR,
 NPMIN, NPMA \emptyset , INPMA \emptyset ,
 NSUC, NTRIAL, TOLREL, TOLABS, XRMIN, XRMAX
 KPASCA, IRAND, INHP, IPRINT

SIGMA returns to the calling program the output parameters

NTRAJ, ISEGBR, KPBR \emptyset , 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})

X \emptyset 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 $f(x)$ 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, X0, H, EPS, DX,
           NTRAJ, ISEGBR, KPBR0, INKPBR,
           NPMIN, NPMAX0, INPMAX,
           NSUC, NTRIAL, TOLREL, TOLABS, XRMIN, XRMAX,
           KPASCA, IRAND, INIP, 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 con-
 tinuation 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 rescal-
 ing, 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 $f(\underline{x}) = \frac{1}{5} x_1^6 - 2.1 x_1^4 + 4x_1^2 + x_1 x_2 + 4x_2^4 - 4x_2^2$ 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 10RIA), 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. 8545) function evaluations.

ACKNOWLEDGEMENT: One of us (F.E.) 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.

```

1.  C
2.  C  MAIA PROGRAM (SAMPLE VERSION)
3.  C  CALLS  SIGMA  VIA THE DRIVER SUBROUTINE  SIGMA1
4.  C
5.  C      DOUBLE PRECISION XC,XMIN,FMIN
6.  C
7.  C      DIMENSION XC(2),XMIN(2)
8.  C
9.  C  TEST PROBLEM DATA
10. C
11. C  PROBLEM DIMENSION
12. C      N = 2
13. C
14. C  INITIAL POINT
15. C      XC(1) = 0.00
16. C      XC(2) = 0.00
17. C
18. C  SET INPUT PARAMETERS
19. C      NSUC = 7
20. C      IPRINT = 0
21. C
22. C  CALL DRIVER SUBROUTINE  SIGMA1
23. C      CALL SIGMA1(N,XC,NSUC,IPRINT,XMIN,FMIN,NFEV,IOUT)
24. C
25. C      STOP
26. C      END

27. C      DOUBLE PRECISION FUNCTION FUNCT (N,X)
28. C
29. C  COMPUTES THE VALUE AT X  OF THE SIX-HUMP CAMEL FUNCTION
30. C
31. C      DOUBLE PRECISION X,XX,YY
32. C      DIMENSION X(N)
33. C      XX = X(1)*X(1)
34. C      YY = X(2)*X(2)
35. C      FUNCT = ((XX/3.00-2.100)+XX+4.00)*XX+X(1)*X(2)
36. C      +4.00*(YY-1.00)*YY
37. C      RETURN
38. C      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 equations, 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 Semi-numerical 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(D,)

```
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 MEAN-
12.     C ING 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 PERFORM-
45.     C           ING 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 ISTCPT
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.     C   DOUBLE PRECISION X0,XMIN,FMIN
```

```

56.      DOUBLE PRECISION DX, EPS, H, TOLABS, TOLREL
57.      DOUBLE PRECISION VRMAX, VRMIN, XRMAX, XRMIN
58.      DIMENSION XO(N), XMIN(N)
59.      DIMENSION XRMIN(100), XRMAX(100)
60.      DATA VRMIN, VRMAX /-1.04, 1.04/
61.      DATA NTRILD/50/
62.      C
63.      H = 1.0-10
64.      EPS = 1.00
65.      DX = 1.0-9
66.      IRAND = 0
67.      NTRAJ = 0
68.      ISEGBR = 0
69.      INKPBR = 0
70.      KPBRQ = 0
71.      NPMIN = 10
72.      NPMAX0 = 100
73.      INPMAX = 50
74.      NTRIAL = MAX0(NTRILD, 5+NSUC)
75.      TOLREL = 1.0-3
76.      TOLABS = 1.0-6
77.      KPASCA = 10
78.      IF(N.GT.5)KPASCA = 300
79.      INHP = 1
80.      DO 1 IX = 1, N
81.          XRMIN(IX) = VRMIN
82.          XRMAX(IX) = VRMAX
83.      1 CONTINUE
84.      C
85.      CALL SIGMA ( N, XO, H, EPS, DX,
86.      1          NTRAJ, ISEGBR, KPBRQ, INKPBR,
87.      2          NPMIN, NPMAX0, INPMAX,
88.      3          NSUC, NTRIAL, TOLREL, TOLABS, XRMIN, XRMAX,
89.      4          KPASCA, IRAND, INHP, IPRINT,
90.      5          XMIN, FMIN, NFEV, IOUT )
91.      C
92.      RETURN
93.      END

94.      SUBROUTINE SIGMA ( N, XO, H, EPS, DX,
95.      1          NTRAJ, ISEGBR, KPBRQ, INKPBR,
96.      2          NPMIN, NPMAX0, INPMAX,
97.      3          NSUC, NTRIAL, TOLREL, TOLABS, XRMIN, XRMAX,
98.      4          KPASCA, IRAND, INHP, IPRINT,
99.      5          XMIN, FMIN, NFEV, IOUT )
100.     C
101.     C THE SUBROUTINE SIGMA IS THE PRINCIPAL SUBROUTINE OF THE PACKAGE
102.     C SIGMA, WHICH ATTEMPTS TO FIND A GLOBAL MINIMIZER OF A REAL VALUED
103.     C FUNCTION  $F(X) = F(X_1, \dots, X_N)$  OF N REAL VARIABLES  $X_1, \dots, X_N$ .
104.     C THE ALGORITHM AND THE PACKAGE ARE DESCRIBED IN DETAIL IN THE TM (

```

105. C PAPERS PUBLISHED IN THE SAME ISSUE OF THE A.C.M. TRANSACTIONS ON
106. C MATHEMATICAL SOFTWARE, BOTH BY
107. C F. ALUFFI-PENTINI, V. PARISI, F. ZIRILLI.
108. C (1) A GLOBAL MINIMIZATION ALGORITHM USING STOCHASTIC DIFFERENTIAL
109. C EQUATIONS
110. C (2) ALGORITHM SIGMA, A STOCHASTIC-INTEGRATION GLOBAL MINIMIZATION
111. C ALGORITHM.
112. C THE SOFTWARE IMPLEMENTATION AND ITS USAGE ARE DESCRIBED IN (2).
113. C
114. C METHOD
115. C
116. C A GLOBAL MINIMIZER OF $F(X)$ IS SOUGHT BY MONITORING THE VALUES OF
117. C F ALONG TRAJECTORIES GENERATED BY A SUITABLE (STOCHASTIC) DISCRE-
118. C TIZATION OF A FIRST-ORDER STOCHASTIC DIFFERENTIAL EQUATION INSPIRED
119. C BY STATISTICAL MECHANICS. STARTING FROM AN INITIAL POINT X_0 ,
120. C X IS UPDATED BY THE (STOCHASTIC) DISCRETIZATION STEP
121. C $X = X + DX1 + DX2$
122. C WHERE $DX1 = -H + GAM$ (FIRST HALF-STEP)
123. C $DX2 = EPS * SQRT(H) * U$ (SECOND HALF-STEP)
124. C AND H IS THE TIME-INTEGRATION STEPLENGTH,
125. C GAM/H IS COMPUTED AS A FINITE-DIFFERENCE APPROXIMATION TO THE
126. C DIRECTIONAL DERIVATIVE OF F ALONG AN ISOTROPICALLY RANDOM
127. C DIRECTION,
128. C EPS IS A POSITIVE "NOISE" COEFFICIENT, AND
129. C U IS A RANDOM SAMPLE FROM AN M -DIMENSIONAL GAUSSIAN DISTRIBUTION.
130. C WE CONSIDER THE SIMULTANEOUS EVOLUTION OF A GIVEN FIXED NUMBER
131. C $NTRAJ$ OF TRAJECTORIES DURING AN OBSERVATION PERIOD IN WHICH FOR
132. C EACH TRAJECTORY EPS IS FIXED WHILE H AND THE SPATIAL DISCRETI-
133. C ZATION INCREMENT DX FOR COMPUTING GAM ARE AUTOMATICALLY
134. C ADJUSTED BY THE ALGORITHM.
135. C AFTER EVERY OBSERVATION PERIOD ONE OF THE TRAJECTORIES IS DISCARDED,
136. C ALL OTHER TRAJECTORIES CONTINUE UNPERTURBED, AND ONE OF THEM IS SE-
137. C LECTED FOR BRANCHING, I.E. GENERATING ALSO A SECOND PERTURBED CONTI-
138. C NUATION, WITH DIFFERENT STARTING EPS AND DX (AND THE SAME
139. C "PAST HISTORY" OF THE FIRST).
140. C THE SET OF SIMULTANEOUS TRAJECTORIES IS CONSIDERED A SINGLE TRIAL,
141. C AND THE COMPLETE ALGORITHM IS A SET OF REPEATED TRIALS.
142. C A TRIAL IS STOPPED, AT THE END OF AN OBSERVATION PERIOD, AND AFTER
143. C HAVING DISCARDED THE WORST TRAJECTORY, IF ALL THE FINAL VALUES OF
144. C F FOR THE REMAINING TRAJECTORIES ARE EQUAL (WITHIN NUMERICAL TOL-
145. C ERANCES, AND POSSIBLY AT DIFFERENT POINTS X) TO THEIR MINIMUM
146. C VALUE $FTFMIN$ ("UNIFORM STOP AT THE LEVEL $FTFMIN$ ").
147. C AN UNIFORM STOP IS CONSIDERED SUCCESSFUL ONLY IF THE FINAL VALUE
148. C $FTFMIN$ IS (NUMERICALLY) EQUAL TO THE CURRENT BEST MINIMUM $FOP1$
149. C FOUND SO FAR FROM ALGORITHM START.
150. C A TRIAL IS ALSO ANYWAY STOPPED (UNSUCCESSFULLY) IF A GIVEN MAXIMUM
151. C NUMBER $NPMAX$ OF OBSERVATION PERIODS HAS ELAPSED.
152. C TRIALS ARE REPEATED WITH DIFFERENT OPERATING CONDITIONS (INITIAL
153. C POINT, MAX TRIAL LENGTH $NPMAX$, SEED OF NOISE GENERATOR, POLICY
154. C FOR CHOOSING THE STARTING EPS FOR THE PERTURBED CONTINUATION,
155. C AND TRIAL-START VALUE OF EPS).
156. C THE ALGORITHM IS STOPPED, AT THE END OF A TRIAL, IF A GIVEN NUMBER
157. C $NSUC$ OF UNIFORM STOPS AT THE CURRENT $FOP1$ LEVEL HAS BEEN REACHED,
158. C OR ANYWAY IF A GIVEN MAXIMUM NUMBER $NTRIAL$ OF TRIALS HAS BEEN
159. C REACHED.
160. C SUCCESS IS CLAIMED IF AT LEAST ONE UNIFORM STOP OCCURRED AT THE
161. C FINAL VALUE OF $FOP1$.

```

162. C
163. C CALL STATEMENT
164. C
165. C THE CALL STATEMENT IS
166. C CALL SIGMA ( N, XO, H, EPS, DX,
167. C          NTRAJ, ISEGBR, KPBR0, INKPBR,
168. C          NPMIN, NPMAX0, INPMAX,
169. C          NSUC, NTRIAL, TOLREL, TOLABS, XRMIN, XRMAX,
170. C          KPA&CA, IRAND, INHP, IPRINT,
171. C          XMIN, FMIN, MFEV, IOUT )
172. C
173. C CALL PARAMETERS
174. C
175. C INPUT PARAMETERS ARE THOSE IN LINES 1,3,4,5 OF THE CALL STATEMENT.
176. C INPUT-OUTPUT PARAMETERS ARE THOSE IN LINE 2.
177. C OUTPUT PARAMETERS ARE THOSE IN LINE 6.
178. C NOTE THAT A NUMBER OF OTHER (INTERNAL) PARAMETERS CAN BE OBTAINED
179. C BY MEANS OF THE USER-SUPPLIED OUTPUT SUBROUTINES PMSIG, PTRIAL,
180. C AND PTKSUC, WHICH ARE DESCRIBED BELOW.
181. C
182. C DESCRIPTION OF THE CALL PARAMETERS
183. C
184. C N      IS THE PROBLEM DIMENSION (NUMBER OF VARIABLES)
185. C XO     IS AN N-VECTOR CONTAINING THE INITIAL VALUES OF THE
186. C        X-VARIABLES
187. C H     IS THE INITIAL VALUE OF THE TIME-INTEGRATION STEPLENGTH.
188. C EPS   IS THE INITIAL VALUE OF THE NOISE COEFFICIENT.
189. C DX    IS THE INITIAL VALUE OF THE MAGNITUDE OF THE DISCRETIZATION
190. C        INCREMENT FOR COMPUTING THE FINITE-DIFFERENCE DERIVATIVES.
191. C NTRAJ IS THE NUMBER OF SIMULTANEOUS TRAJECTORIES.
192. C        (NOTE HOWEVER THAT IF THE INPUT VALUE IS ZERO, NTRAJ IS
193. C        SET TO A DEFAULT VALUE (NTRAJ = 7), AND IF THE INPUT VALUE
194. C        IS OTHERWISE OUTSIDE THE INTERVAL (3,20) NTRAJ IS SET TO
195. C        THE NEAREST EXTREME VALUE).
196. C ISEGBR, KPBR0, INKPBR DETERMINE, AT THE END OF AN OBSERVATION
197. C        PERIOD, WHICH ONE OF THE SIMULTANEOUS TRAJECTORIES
198. C        IS TO BE BRANCHED, AS FOLLOWS.
199. C        BRANCHING IS NORMALLY PERFORMED ON THE TRAJECTORY WHICH
200. C        OCCUPIES THE PLACE ISEGBR IN THE TRAJECTORY SELECTION OR-
201. C        DERING, EXCEPT AT (THE END OF) EXCEPTIONAL OBSERVATION
202. C        PERIODS, WHERE THE FIRST TRAJECTORY IN THE ORDERING IS
203. C        BRANCHED. EXCEPTIONAL BRANCHING OCCURS AT THE OBSERVATION
204. C        PERIODS NUMBERED KP = KPBR0 + J*INKPBR, (J = 1,2,3,...).
205. C        THEREFORE ISEGBR SELECTS THE LEVEL (IN THE ORDERING) AT
206. C        WHICH NORMAL BRANCHING OCCURS, WHILE KPBR0 AND INKPBR
207. C        SELECT THE FIRST OCCURRENCE AND THE REPETITION FREQUENCY
208. C        OF THE EXCEPTIONAL OBSERVATION PERIODS.
209. C        (NOTE HOWEVER THAT IF ONE OF THE INPUT VALUES IS ZERO,
210. C        THE CORRESPONDING VARIABLE IS SET TO A DEFAULT VALUE
211. C        ISEGBR = INT((1+NTRAJ)/2), INKPBR = 10, KPBR0 = 3.
212. C        IF THE INPUT VALUE FOR ISEGBR IS OTHERWISE OUTSIDE THE
213. C        INTERVAL (1,NTRAJ), ISEGBR IS SET TO THE NEAREST
214. C        EXTREME VALUE, AND IF KPBR0 HAS A VALUE NOT INSIDE THE
215. C        INTERVAL (1,INKPBR), IT IS ASSIGNED THE SAME VALUE
216. C        MODULO INKPBR).
217. C NPMIN  IS THE MINIMUM DURATION OF A TRIAL, I.E. THE MINIMUM
218. C        NUMBER OF OBSERVATION PERIODS THAT SHOULD ELAPSE BEFORE

```


219. C STARTING TO CHECK THE TRIAL STOPPING CRITERIA.
220. C NPMAX0 IS THE MAXIMUM DURATION OF THE FIRST TRIAL, I.E. THE
221. C VALUE, FOR THE FIRST TRIAL, OF MAXIMUM ACCEPTABLE
222. C NUMBER NPMAX OF OBSERVATION PERIODS IN A TRIAL.
223. C INPMAX IS THE INCREMENT FOR NPMAX, WHEN NPMAX IS VARIED
224. C FROM ONE TRIAL TO THE FOLLOWING ONE.
225. C NSUC IS THE NUMBER OF SUCCESSFUL TRIALS (WITH THE SAME FINAL
226. C VALUE FOPT) AFTER WHICH THE COMPUTATION IS STOPPED.
227. C TOLREL AND TOLABS ARE THE RELATIVE AND ABSOLUTE TOLERANCES
228. C FOR STOPPING A SINGLE TRIAL.
229. C XRMIN, XRMAX ARE N-VECTORS DEFINING THE ADMISSIBLE REGION FOR
230. C THE X-VALUES, WITHIN WHICH THE FUNCTION VALUES CAN BE
231. C SAFELY COMPUTED.
232. C KPASCA IS THE MINIMUM NUMBER OF TRAJECTORY SEGMENTS THAT SHOULD
233. C ELAPSE BEFORE THE RESCALING PROCEDURES ARE ACTIVATED.
234. C IRAND IS A CONTROL INDEX FOR THE INITIALIZATION OF THE RANDOM
235. C NUMBER GENERATOR.
236. C IRAND.GT.0 THE GENERATOR IS INITIALIZED, BEFORE STAR-
237. C TING THE TRIAL KT, WITH SEED IRAND+KT-1
238. C IRAND.LE.0 THE GENERATOR IS INITIALIZED (WITH SEED 0)
239. C ONLY AT THE FIRST CALL OF SIGMA
240. C INHP IS A CONTROL INDEX FOR SELECTING THE NUMBER NHP OF TYPE-
241. C INTEGRATION STEPS FOR OBSERVATION PERIOD KP (DURATION OF
242. C TRIAL KP) AS FOLLOWS (LOG IS BASE 2)
243. C INHP=1 NHP = 1 + INT(LOG(KP)) ('SHORT' DURATION)
244. C INHP=2 NHP = INT(SQRT(KP)) ('MEDIUM' DURATION)
245. C INHP=3 NHP = KP ('LONG' DURATION)
246. C IPRINT IS AN INDEX USED TO CONTROL THE AMOUNT OF PRINTED OUTPUT
247. C BY CONTROLLING THE CALLS TO THE USER-SUPPLIED OUTPUT SUB-
248. C ROUTINES PTSEG (END-OF-SEGMENT OUTPUT), PTRIAL (END-
249. C OF-TRIAL OUTPUT), AND PTKSUC (END-OF-TRIAL OUTPUT RELATED
250. C TO THE COUNT OF SUCCESSFUL TRIALS), WHICH ARE DESCRIBED
251. C BELOW.
252. C IPRINT.LT.0 NO CALL TO THE OUTPUT SUBROUTINES
253. C IPRINT.EQ.0 CALL ONLY PTRIAL AND PTKSUC
254. C IPRINT.GT.0 CALL ALL OUTPUT SUBROUTINES.
255. C XMIN IS AN N-VECTOR CONTAINING THE COORDINATES OF THE POINT
256. C (OR POSSIBLY ONE OF THE POINTS) WHERE THE FINAL VALUE FMIN
257. C OF FOPT WAS FOUND.
258. C FMIN IS THE FINAL VALUE OF THE BEST CURRENT MINIMUM FUNCTION
259. C VALUE FOPT.
260. C NFEV IS THE TOTAL NUMBER OF FUNCTION EVALUATION (INCLUDING
261. C THOSE USED FOR THE COMPUTATION OF DERIVATIVES, AND FOR
262. C THE REJECTED TIME-INTEGRATION STEPS).
263. C IOUT IS THE INDICATOR OF THE STOPPING CONDITIONS, AS FOLLOWS
264. C IF IOUT = -99 A FATAL ERROR WAS DETECTED WHEN PERFORM-
265. C ING SOME PRELIMINARY CHECKING OF THE INPUT DATA, AND
266. C THE ALGORITHM WAS NOT EVEN STARTED
267. C OTHERWISE THE ALGORITHM WAS STARTED, AND THE VALUE OF
268. C IOUT IS THE FINAL VALUE OF THE INTERNAL PARAMETER ISTOPT
269. C (AN OUTPUT INDICATOR OF THE USER-SUPPLIED SUBROUTINE
270. C PTRIAL, DESCRIBED BELOW).
271. C SUCCESS IS CLAIMED BY THE ALGORITHM IF IOUT .GT. 0,
272. C I.E. IF AT LEAST ONE OF THE TRIALS STOPPED WITH A POSITIVE
273. C VALUE OF THE TRIAL STOPPING INDICATOR ISTOPT (AN OUTPUT
274. C INDICATOR OF THE USER-SUPPLIED SUBROUTINE PTRIAL,
275. C DESCRIBED BELOW), AND NO LOWER VALUE FOR FOPT WAS FOUND

```

276. C           IN THE FOLLOWING TRIALS.
277. C
278. C USER-SUPPLIED SUBPROGRAMS
279. C
280. C THE USER MUST PROVIDE THE FUNCTION FUNCT TO COMPUTE F(X),
281. C AND THE THREE OUTPUT SUBROUTINE PTSEG, PTRIAL, PTKSUC .
282. C THE CALLS TO THE OUTPUT SUBROUTINES ARE CONTROLLED BY IPRINT
283. C (INPUT PARAMETER TO SIGMA).
284. C A USER NOT INTERESTED IN USING ANY ONE OF THE OUTPUT SUBROUTINES
285. C MUST PROVIDE A DUMMY SUBROUTINE (WITH RETURN AS THE ONLY
286. C EXECUTABLE STATEMENT) TO AVOID UNRESOLVED REFERENCES.
287. C IN THE FOLLOWING DESCRIPTION ALL NON-INTEGGER ARGUMENTS ARE
288. C DOUBLE PRECISION (INTEGER ARGUMENTS ARE INDICATED BY MEANS OF THE
289. C FORTRAN IMPLICIT TYPE DEFINITION CONVENTION).
290. C
291. C THE FUNCTION FUNCT
292. C
293. C FUNCT MUST RETURN AS ITS VALUE THE VALUE AT X OF THE FUNCTION
294. C TO BE MINIMIZED
295. C THE DEFINITION STATEMENT IS
296. C     DOUBLE PRECISION FUNCTION FUNCT (N, X)
297. C     WHERE
298. C     N IS THE (INPUT) DIMENSION OF THE PROBLEM.
299. C     X IS THE (INPUT) N-VECTOR CONTAINING THE COORDINATES OF THE
300. C     POINT X WHERE THE FUNCTION IS TO BE COMPUTED.
301. C
302. C THE SUBROUTINE PTSEG
303. C
304. C PTSEG IS CALLED (IF IPRINT.GT.0) AT THE END OF EVERY OBSER-
305. C VATION PERIOD.
306. C THE DEFINITION STATEMENT IS
307. C     SUBROUTINE PTSEG ( N, XPFMIN, FPFMIN, FPFMAX,
308. C                     KP, NFEV, IPRINT )
309. C     WHERE
310. C     N IS THE (INPUT) DIMENSION OF THE PROBLEM
311. C     FPFMIN, FPFMAX ARE RESPECTIVELY THE MINIMUM AND THE MAXIMUM
312. C     AMONG THE VALUES OF F(X) OBTAINED AT THE FINAL POINTS OF
313. C     THE TRAJECTORY SEGMENTS OF THE (JUST ELAPSED) OBSERVATION
314. C     PERIOD KP.
315. C     XPFMIN IS AN N-VECTOR CONTAINING THE COORDINATES OF THE
316. C     (FINAL) POINT (OR POSSIBLY ONE OF THE POINTS) WHERE
317. C     FPFMIN WAS OBTAINED.
318. C     KP IS THE TOTAL NUMBER OF ELAPSED OBSERVATION PERIODS IN
319. C     THE CURRENT TRIAL.
320. C     NFEV IS THE TOTAL NUMBER OF FUNCTION EVALUATIONS PERFORMED
321. C     FROM ALGORITHM START.
322. C
323. C THE SUBROUTINE PTRIAL
324. C
325. C PTRIAL IS CALLED (IF IPRINT.GE.0) AT THE END OF EVERY TRIAL.
326. C THE DEFINITION STATEMENT IS
327. C     SUBROUTINE PTRIAL ( N, XOPT, FOPT,
328. C                     FTFMIN, FTFMAX, FTFOPT,
329. C                     ISTOP, ISTOPT, NFEV, KP, IPRINT )
330. C     WHERE
331. C     N IS THE (INPUT) DIMENSION OF THE PROBLEM.
332. C     XOPT IS AN N-VECTOR CONTAINING THE COORDINATES OF THE

```

333. C POINT (OR POSSIBLY ONE OF THE POINTS) WHERE THE CURRENT
 334. C MINIMUM FOPT WAS OBTAINED.
 335. C FOPT IS THE CURRENT BEST MINIMUM VALUE FOUND FOR F FROM
 336. C ALGORITHM START (FOPT IS UPDATED WHENEVER A FUNCTION
 337. C VALUE IS COMPUTED).
 338. C FTFMIN, FTFMAX ARE RESPECTIVELY THE MINIMUM AND THE MAXIMUM
 339. C AMONG THE VALUES OF F(X) OBTAINED AT THE FINAL POINTS OF
 340. C THE LAST TRAJECTORY SEGMENTS OF THE CURRENT TRIAL.
 341. C FTFOPT IS THE CURRENT MINIMUM VALUE OF FTFMIN AMONG THE
 342. C TRIALS WHICH DID NOT STOP FOR REACHING THE MAXIMUM ALLOWED
 343. C NUMBER OF SEGMENTS (STOPPING INDICATOR ISTOP = 0, SEE
 344. C BELOW). FTFOPT IS USED BY SIGMA TO COMPUTE NSUC (INPUT
 345. C PARAMETER TO THE OUTPUT SUBROUTINE PTKSUC, SEE BELOW).
 346. C KP IS THE TOTAL NUMBER OF ELAPSED OBSERVATION PERIODS IN
 347. C THE CURRENT TRIAL.
 348. C NFEV IS THE TOTAL NUMBER OF FUNCTION EVALUATIONS PERFORMED
 349. C FROM ALGORITHM START.
 350. C ISTOP IS THE INDICATOR OF THE STOPPING CONDITION OF THE TRIAL,
 351. C AS FOLLOWS
 352. C ISTOP = 0
 353. C THE MAXIMUM NUMBER NPMAX OF OBSERVATION PERIODS HAS
 354. C BEEN REACHED.
 355. C ISTOP.NE.0
 356. C ALL THE END-OF-SEGMENT VALUES OF F(X), (EXCEPT FOR THE
 357. C JUST DISCARDED SEGMENT) ARE CLOSE ENOUGH TO THEIR COMMON
 358. C MINIMUM VALUE FTFMIN, WITH RESPECT TO AN ABSOLUTE OR
 359. C RELATIVE DIFFERENCE CRITERION, TO BE CONSIDERED NUMERI-
 360. C CALLY EQUAL.
 361. C THE ABSOLUTE VALUE AND THE SIGN OF ISTOP HAVE THE
 362. C FOLLOWING MEANING.
 363. C THE ABSOLUTE VALUE INDICATES WHICH DIFFERENCE
 364. C CRITERION WAS SATISFIED
 365. C 1 RELATIVE DIFFERENCE CRITERION SATISFIED
 366. C 2 ABSOLUTE DIFFERENCE CRITERION SATISFIED
 367. C 3 BOTH CRITERIA SATISFIED
 368. C THE SIGN OF ISTOP INDICATES THE RELATIONSHIP BETWEEN
 369. C THE END-OF-TRIAL VALUE FTFMIN AND THE CURRENT
 370. C BEST MINIMUM VALUE FOPT (WHICH IS UPDATED WHEN-
 371. C EVER A FUNCTION VALUE IS COMPUTED
 372. C ISTOP.GT.0
 373. C FTFMIN IS NUMERICALLY EQUAL (W.R.T. AT LEAST
 374. C ONE OF THE ABOVE DIFFERENCE CRITERIA) TO FOPT
 375. C ISTOP.LT.0
 376. C FTFMIN IS NOT EVEN NUMERICALLY EQUAL TO FOPT
 377. C (AND THEREFORE CANNOT BE CONSIDERED AS AN
 378. C ACCEPTABLE GLOBAL MINIMUM).
 379. C ISTOPT IS THE VALUE OF THE TRIAL STOPPING INDICATOR ISTOP
 380. C CORRESPONDING TO THE (CURRENT OR PAST) TRIAL WHERE FTFOPT
 381. C WAS OBTAINED, WITH THE SIGN WHICH IS UPDATED ACCORDING TO
 382. C THE COMPARISON BETWEEN FTFOPT AND THE PRESENT VALUE OF
 383. C FOPT, AS DESCRIBED ABOVE.
 384. C THE FINAL VALUE OF ISTOP IS RETURNED BY SIGMA AS THE VALUE
 385. C OF THE OUTPUT INDICATOR IOUT OF THE ALGORITHM STOPPING CON-
 386. C DITIONS (WHENEVER THE ALGORITHM WAS STARTED, IOUT.NE.-99,
 387. C SEE ABOVE).
 388. C
 389. C THE SUBROUTINE PINSUC

```

390. C
391. C PTKSUC IS CALLED ONLY AT THE END OF EVERY SUCCESSFUL TRIAL
392. C SUCH THAT AN INCREMENT OCCURRED IN THE VALUE KSUC OF THE
393. C MAXIMUM NUMBER OF SUCCESSFUL TRIALS AT THE SAME (CURRENT OR
394. C PAST) VALUE OF FORT . A CALL TO PTKSUC THEREFORE PROVIDE :
395. C THE USER WITH THE OPERATIONALLY INTERESTING INFORMATION THAT
396. C A FINAL SUCCESS CLAIM WOULD HAVE TAKEN PLACE, IF NSUC (INPUT
397. C PARAMETER TO SIGMA ) HAD BEEN GIVEN A LOWER VALUE, EQUAL TO
398. C THE CURRENT KSUC .
399. C PTKSUC IS CALLED ONLY IF IPRINT.GE.0 AND KSUC.LT.NSUC .
400. C THE DEFINITION STATEMENT IS
401. C SUBROUTINE PTKSUC ( KSUC )
402. C WHERE KSUC IS THE INTEGER VARIABLE (1 .LE. KSUC .LE. NSUC)
403. C DEFINED ABOVE.
404. C
405. C DOUBLE PRECISION XQ,H,EPS,DX,TOLREL,TOLABS
406. C DOUBLE PRECISION XRMIN,XRMAX,XMIN,FMIN
407. C DOUBLE PRECISION EPSAG,EPSAP,EPSC
408. C DOUBLE PRECISION EPSMAX,EPSR,F,FOPT,FTFMAX
409. C DOUBLE PRECISION FTFMIN,FTFOPT
410. C
411. C DOUBLE PRECISION X,HC,DXC,VMVT,EPSCO,VMCOR,VCOR
412. C DOUBLE PRECISION XRMIC,XRMAC,XOPT,FOPTC
413. C
414. C DOUBLE PRECISION DIST,BIAS,GRAGRA,GRA
415. C
416. C DIMENSION XQ(N),XMBN(N),XRMIN(N),XRMAX(N)
417. C
418. C COMMON /DINCOM/ X(100,20),HC(20),DXC(20),VMVT(20,19),EPSCO(20),
419. C 1 VMCOR(20),VCOR(20),XRMIC(100),XRMAC(100),XOPT(100),FOPTC ,
420. C 2 IE(20),ISVT(20,19),KGEN,KTIM,NDIM,NTRAJC,NTRAJR,
421. C 3 ISEGB,INKPBC,KPBRO,MCF,IFEP,INHPC
422. C
423. C COMMON /SCALE/ DIST(10,10,20),BIAS(10,20),GRAGRA(10,10,20),
424. C 1 GRA(10,20),NGRA(20),LSCA,IBSCA,MX,NORD
425. C
426. C DATA FOR THE VARIATION OF NOISE COEFFICIENT
427. C
428. C DATA EPSR/1.04/,EPSAP/10.00/,EPSAG/1.03/
429. C DATA EPSMAX/1.015/
430. C
431. C IFEP = 1
432. C
433. C INITIALIZE COMMON AREA /DINCOM/
434. C
435. C CALL INIT(N,XQ,H,ERS,DX,IRAND,F,
436. C 1 NTRAJ,ISEGB,INKPBR,KPBRO,INHPC,IFEP,XRMIN,XRMAX,IOUT)
437. C
438. C CHECK PARAMETER VALUES
439. C
440. C IF(NPMIN.LE.0.OR.NPMAXO.LT.0.OR.INPMAX.LE.0.OR.
441. C 1 NSUC.LE.0.OR.NTRIAL.LE.0)IOUT = -99
442. C IF (IOUT.EQ.(-99))RETURN
443. C
444. C INITIALIZE VARIABLES
445. C
446. C EPSC = ERS

```

```

447.      NPMAX = NPMAX0
448.      ISTOP = 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
462.      IS = 0
463.      IF(IRAND.GT.0)IS = IRAND+IC-1
464.      C
465.      C INITIALIZE TRIAL
466.      C
467.      IF(IC.GT.1.AND.IC.LE.ICCOM)CALL REINIT(N,X0,EPSC,IS,F,IFEF)
468.      IF(IC.GT.ICCOM)CALL REINIT(N,XMIN,EPSC,IS,FOP1,IFEF)
469.      C
470.      FTFMIN = F
471.      FTFMAX = F
472.      NFEV = NFEV+1
473.      C
474.      C PRINT INITIAL CONDITIONS OF TRIAL
475.      C
476.      IF(IPRINT.GT.0)CALL PTSEG(N,X0,FTFMIN,FTFMAX,0,NFEV)
477.      C
478.      C
479.      C DEACTIVATE SCALING
480.      C
481.      IF(KPASCA.GT.NPMAX.OR.N.LE.1)CALL NOSCA
482.      C
483.      C INITIALIZE COMMON AREA /SCALE/
484.      C
485.      IF(KPASCA.LE.NPMAX.AND.N.GT.1)CALL INISCA(N,NTRAJ)
486.      C
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
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
502.      C COMPUTE INDICATOR OF TRIAL STOPPING CONDITIONS
503.      C UPDATE BEST CURRENT VALUES OF TRIAL STOPPING INDICATOR AND

```

```

504. C OF FUNCTION F(X) AT TRIAL STOP
505. C
506. IF((FTFMIN.GT.FTFOPT.OR.ISTOP.EQ.0).AND.ISTOPT.NE.0)GO TO 10
507. IF((ITOLCH(FTFOPT,FTFMIN,TOLREL,TOLABS).EQ.0)NTES = NSUC
508. C
509. FTFOPT = FTFMIN
510. ISTOPT = ISTOP
511. 10 CONTINUE
512. ISTOPT = IABS(ISTOPT)
513. CALL RCLOPT(N,XMIN,FOPT)
514. IF((ITOLCH(FTFOPT,FOPT,TOLREL,TOLABS).EQ.0)ISTOPT = -ISTOP 1
515. IF((ITOLCH(FTFMIN,FOPT,TOLREL,TOLABS).EQ.0)ISTOP = -ISTOP
516. C
517. C END-OF-TRIAL PRINT OUT
518. C
519. IF((IPRINT.GE.0)
520. 1 CALL PTRIAL (N, XOPT, FOPT,
521. 2 FTFMIN, FTFMAX, FTFOPT,
522. 3 ISJOP, ISTOPT, NFEV, KP, IPRINT )
523. C
524. C UPDATE INITIAL VALUE OF NOISE COEFFICIENT FOR NEXT TRIAL
525. C
526. IF (ISTOP.EQ.0)EPSC = EPSC/EPSR
527. IF(ISTOP.GT.0)ERSC = EPSC*EPSAG
528. IF(ISTOP.LT.0.AND.IC.LE.ICCOM)EPSC = EPSC*EP6AP
529. IF(ISTOP.LT.0.AND.IC.GT.ICCOM)EPSC = EPSC/EP6R
530. C
531. C UPDATE OPERATING CONDITIONS FOR SELECTING (IN THE NEXT TRIAL)
532. C THE STARTING VALUE OF THE NOISE COEFFICIENT OF THE NEW TRAJECTORY
533. C AFTER BRANCHING
534. C
535. IF (ISTOP.EQ.0)IFEP = 1
536. IF (ISTOP.NE.0)IFEP = 2
537. C
538. C UPDATE, PRINT, AND CHECK TRIAL STOPPING CONDITIONS
539. C
540. IF(ISTOP.GT.0.AND.ISTOPT.GT.0)NTES = NTES-1
541. IF(NTES.LE.0.OR.ISTOPT.LE.0.OR.ISTOP.LE.0.OR.NTES.GT.ICTS.OR.
542. 1 IPRINT.LT.0) GO TO 20
543. KSUC = NSUC-ICTS
544. CALL PTKSUC(KSUC)
545. ICTS = NTES-1
546. 20 CONTINUE
547. IF(NTES.LE.0.AND.ISTOPT.GT.0.AND.ISTOP.GT.0)GO TO 40
548. C
549. C CONSTRAIN NOISE COEFFICIENT WITHIN BOUNDS
550. C
551. EPSC = DMINT(EPSC,EP5MAX)
552. IF(EPSC.LE.0.00)EPSC = 1.00
553. 30 CONTINUE
554. C END OF SERIES OF TRIALS
555. C
556. 40 CONTINUE
557. C
558. C INDICATOR OF STOPPING CONDITIONS
559. C
560. IOUT = ISTOPT

```

```

1206.      CALL DERGEN(NDIM,X,FV,DX,W,DFDX)
1207.      C
1208.      C TRY AGAIN THE FIRST HALF-STEP
1209.      DO 40 IC = 1,NDIM
1210.          XP(IC) = X(IC)-H*W(IC)*DFDX*DBLE(FLOAT(NDIM))
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,JOLABS).GT.0)DX = DX*RCB
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
1239.          XP(IC) = XP(IC)+EPSR*W(IC)
1240.      60  CONTINUE
1241.          F = FUNCTO(NDIM,XP)
1242.      C
1243.      C ACCEPT OR REJECT THE COMPLETE STEP
1244.      IF (F-FV.LE.EPS*ERS*STF) 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
1252.          X(IC) = XP(IC)
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 DERCEM ) TO COMPUTE THE FORWARD-
1151. C (OR CENTRAL-) DIFFERENCES DIRECTIONAL DERIVATIVE GAM/W
1152. C - CALLS THE SUBROUTINE NEWM TO ACCEPT OR REJECT THE FIRST HALF-
1153. C STEP AND OBTAIN AN UPDATED VALUE FOR H
1154. C - CALLS THE SUBROUTINE BUMSCA TO UPDATE THE CUMULATED SCALING DATA
1155. C - UPDATES THE SPATIAL DISCRETIZATION INCRMENT 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. C      DOUBLE PRECISION X,M,EPS,DX,F
1160. C      DOUBLE PRECISION DEDX,DFDXV,DXMAX,DXMIN
1161. C      DOUBLE PRECISION EPSR,FS,FV,FVS,MMINS,MR,MS
1162. C      DOUBLE PRECISION RCD,RDX,STF,TOLABS,TOLRA,TOLRI
1163. C      DOUBLE PRECISION W,XP
1164. C      DOUBLE PRECISION FUNCTO
1165. C      DIMENSION X(NDIM)
1166. C      DIMENSION W(100),XP(100)
1167. C      DATA RDX/1.0-4/
1168. C      DATA DXMIN/1.0-35/
1169. C      DATA DXMAX/1.03/
1170. C      DATA MR/1.0-1/
1171. C      DATA MMINS/1.0-30/
1172. C      DATA STF /100.00/
1173. C      DATA RCD/2.00/
1174. C      DATA TOLRI/1.0-5/
1175. C      DATA TOLRA/1.0-11/
1176. C      DATA TOLABS/0.00/
1177. C
1178. C      IEC = 0
1179. C      FV = F
1180. C      10 CONTINUE
1181. C      20 CONTINUE
1182. C
1183. C TAKE A RANDOM DIRECTION FOR THE DIRECTIONAL DERIVATIVE
1184. C CALL UNIRV(NDIM,W)
1185. C
1186. C COMPUTE FORWARD-DIFFERENCE DERIVATIVE
1187. C CALL DERFOR(NDIM,X,FV,DX,W,DFDX)
1188. C
1189. C TRY THE FIRST HALF-STEP
1190. C DO 30 IC = 1,NDIM
1191. C     XP(IC) = X(IC)-M+W(IC)+DFDX+DBLE(FLOAT(NDIM))
1192. C 30 CONTINUE
1193. C     MS = M
1194. C     F = FUNCTO(NDIM,XP)
1195. C     FVS = FV+DX*DBS(DEDX)
1196. C
1197. C UPDATE STEPLENGTH H AND ACCEPT OR REJECT THE HALF-STEP
1198. C CALL NEWM(KTIM,FVS,F,M,IE,IEC)
1199. C IF(IEC.LE.0)GO TO 50
1200. C IE = IE-1
1201. C IEC = IEC-1
1202. C H = MS
1203. C DFDXV = DFDX
1204. C
1205. C COMPUTE CENTRAL-DIFFERENCES DERIVATIVE

```



```

1100.      1   VMCOR(20),VCOR(20),XRMIN(100),XRMAX(100),XOPT(100),FOPT,
1101.      2   IE(20),ISVT(20,19),KGEN,KTIM,NDIM,NTRAJ,NTRAJR,
1102.      3   ISEGBR,INKPBR,XPBRD,MCF,IFEP,INHP
1103.      C
1104.      1   KTIM = KTIM+1
1105.      C
1106.      C LOOP ON THE SIMULTANEOUS TRAJECTORY SEGMENTS
1107.      1   DO 10 ID = 1,NTRAJ
1108.      C
1109.      C INFORM THE SCALING SUBPROGRAMS (VIA THE COMMON AREA /SCALE/)
1110.      C THAT SCALING OPERATIONS ARE TO BE PERFORMED ON SEGMENT ID .
1111.      1   CALL SEGSCA(ID)
1112.      1   F = VCOR(ID)
1113.      C
1114.      C PERFORM A TIME-INTEGRATION STEP ON SEGMENT ID .
1115.      1   KA = KTIM
1116.      1   MX = NDIM
1117.      1   DO 10 IX = 1,MX
1118.      2   10   XID(IX) = X(IX,ID)
1119.      2   10   CONTINUE
1120.      1   HID = H(ID)
1121.      1   EPSID = ERS(ID)
1122.      1   DXID = DX(ID)
1123.      1   IEID = IE(ID)
1124.      C
1125.      1   CALL SSTEP(KA,MX,XID,HID,EPSID,DXID,IEID,F)
1126.      C
1127.      1   DO 20 IX = 1,MX
1128.      2   20   X(IX,ID) = XID(IX)
1129.      2   20   CONTINUE
1130.      1   H(ID) = HID
1131.      1   EPS(ID) = EPSID
1132.      1   DX(ID) = DXID
1133.      1   IE(ID) = IEID
1134.      1   VCOR(ID) = F
1135.      1   VMCOR(ID) = DMINT(VMCOR(ID),F)
1136.      1   IF(IE.EQ.1)VMCOR(ID) = F
1137.      1   IF(KTIM.EQ.1)IE(ID) = 0
1138.      1   30   CONTINUE
1139.      C
1140.      1   RETURN
1141.      1   END

```

```

1142.      SUBROUTINE SSTEP(KTIM,NDIM,X,H,EPS,DX,IE,F)
1143.      C
1144.      C THE BASIC TIME-INTEGRATION STEP FOR A GIVEN TRAJECTORY IS PERFORMED
1145.      C BY THE SUBROUTINE STEP WHICH
1146.      C - CALLS THE FUNCTION FUNCTO TO COMPUTE THE VALUE OF F
1147.      C - CALLS THE SUBROUTINE UNITRV TO COMPUTE THE RANDOM DIRECTION
1148.      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
3 1053.          40 CONTINUE
1 1054.          50 CONTINUE
1 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
1 1066.          DO 90 ID = 1,NTRAJ
1 1067.          IF(ISVT(1,1).NE.ISVT(ID,1))GO TO 100
1 1068.          90 CONTINUE
1 1069.          IT = 2
1 1070.          GO TO 140
1 1071.          100 CONTINUE
1 1072.          110 CONTINUE
1 1073.          120 CONTINUE
1 1074.          IT = IT1+1
1 1075.          DO 130 ID = 1,NTRAJ
1 1076.          VMVT(ID,IT) = DMIN1(VMVT(ID,IT),VMVT(ID,IT1))
1 1077.          130 CONTINUE
1 1078.          140 CONTINUE
1 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
1 1087.          RETURN
1 1088.          END

```

```

1089.          SUBROUTINE STEP(IX)
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,MID,EPSID,DXID
1098.          DIMENSION XID(100)
1099.          COMMON /DINCOM/ X(100,20),H(20),DX(20),VMVT(20,19),EPS(20),

```

```

1003.      INTEGER FUNCTION IPRECE(ID1, ID2)
1004.      C
1005.      C IPRECE DETERMINES THE PRECEDENCE RELATION BETWEEN TWO TRAJECTORIES
1006.      C BASED ON THEIR CURRENT VALUE OF EPS
1007.      C
1008.      DOUBLE PRECISION X, H, DX, VMVT, EPS, VMCOR, VCOR
1009.      DOUBLE PRECISION XRMIN, XRMAX, XOPT, FOPT
1010.      COMMON /DINCOM/ X(100, 20), H(20), DX(20), VMVT(20, 19), EPS(20),
1011.      1 VMCOR(20), VCOR(20), XRMIN(100), XRMAX(100), XOPT(100), FOPT,
1012.      2 IE(20), ISVT(20, 19), KGEN, KTIM, NDIM, NTRAJ, NTRAJR,
1013.      3 ISEGBR, INKPBR, KPBRO, MCF, IFEP, INHP
1014.      IPRECE = 0
1015.      IF(KGEN.GT.ISEGBR+INKPBR)GO TO 10
1016.      IF(EPS(ID2).LT.EPS(ID1))IPRECE = ID1
1017.      IF(EPS(ID2).GT.EPS(ID1))IPRECE = ID2
1018.      RETURN
1019.      C
1020.      10 CONTINUE
1021.      IF(EPS(ID2).LT.EPS(ID1))IPRECE = ID2
1022.      IF(EPS(ID2).GT.EPS(ID1))IPRECE = ID1
1023.      C
1024.      RETURN
1025.      END

```

```

1026.      SUBROUTINE COMPAS
1027.      C
1028.      C COMPAS TAKES CARE OF THE STORAGE OF PAST HISTORY DATA, DISCARDING
1029.      C ALL DATA NOT NEEDED BY THE ONLY USER OF SUCH DATA, THE FUNCTION
1030.      C IPREC .
1031.      C
1032.      DOUBLE PRECISION X, H, DX, VMVT, EPS, VMCOR, VCOR
1033.      DOUBLE PRECISION XRMIN, XRMAX, XOPT, FOPT
1034.      COMMON /DINCOM/ X(100, 20), H(20), DX(20), VMVT(20, 19), EPS(20),
1035.      1 VMCOR(20), VCOR(20), XRMIN(100), XRMAX(100), XOPT(100), FOPT,
1036.      2 IE(20), ISVT(20, 19), KGEN, KTIM, NDIM, NTRAJ, NTRAJR,
1037.      3 ISEGBR, INKPBR, KPBRO, MCF, IFEP, INHP
1038.      C
1039.      IT1 = 1
1040.      DO 30 IT = 2, NTRAJR
1041.          IT1 = IT-1
1042.          DO 10 ID = 1, NTRAJ
1043.              IF(ISVT(ID, IT).NE.ISVT(ID, IT1))GO TO 20
1044.              10 CONTINUE
1045.              GO TO 120
1046.              20 CONTINUE
1047.              30 CONTINUE
1048.          IT1 = 1
1049.          MCV = 0
1050.          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 IRRREC(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, 19), EPS(20),
      984.      1      VMCOR(20), VCOR(20), XRMIN(100), XRMAX(100), XOPT(100), FOPT,
      985.      2      IE(20), ISVT(20, 19), KGEN, KYIM, NDIM, NTRAJ, NTRAJR,
      986.      3      ISEGBR, INKPBR, KPBR0, NCF, IFEP, INHP
      987.      C
      988.      VM1 = VMVT(ID1, NTRAJR)
      989.      VM2 = VMVT(ID2, NTRAJR)
      990.      DO 10 IIT = 1, NTRAJR
      991.      IT = 1+NTRAJR-IIT
      992.      IF(ISVT(ID1, IT).EQ.ISVT(ID2, IT))GO TO 20
      993.      VM1 = DMIN1(VM1, VMVT(ID1, IT))
      994.      VM2 = DMIN1(VM2, VMVT(ID2, IT))
      995.      10      CONTINUE
      996.      20      CONTINUE
      997.      IPREC = 0
      998.      IF(VM2.LT.VM1)IPREC = 102
      999.      IF(VM2.GT.VM1)IPREC = 101
1000.      C
1001.      RETURN
1002.      END

```

912. END

```
913. SUBROUTINE ORDER(IR,IM,IU)
914. C
915. C ORDER COMPARES THE TRAJECTORIES FROM THE POINT OF VIEW OF PAST
916. C HISTORY (BY CALLING THE FUNCTION IPREC) AND FROM THE POINT OF VIEW
917. C OF THEIR CURRENT VALUE OF EPS (BY CALLING THE FUNCTION IPREC)
918. C AND PROVIDES THE TRAJECTORY ORDERING NEEDED FOR SELECTING THE TRA-
919. C JECTORY WHICH IS TO BE BRANCHED.
920. C
921. C DOUBLE PRECISION X,H,DX,VMVT,EPS,VMCOR,VCOR
922. C DOUBLE PRECISION XRMIN,XRMAX,XOPT,FOPT
923. C COMMON /DINCOM/ X(100,20),H(20),DX(20),VMVT(20,19),EPS(20),
924. C 1 VMCOR(20),VCOR(20),XRMIN(100),XRMAX(100),XOPT(100),FOPT,
925. C 2 IE(20),ISVT(20,19),KGEN,KTIM,NDIM,NTRAJ,NTRAJR,
926. C 3 ISEGBR,INKPBR,KPBRD,NCF,IFEP,INHP
927. C DIMENSION IORD(20)
928. C DATA KP/0/
929. C IR = 0
930. C
931. C ASSIGN INITIAL ORDERING
932. C
933. C 10 CONTINUE
934. C DO 20 I = 1,NTRAJ
935. C IORD(I) = I
936. C 20 CONTINUE
937. C
938. C SORT TRAJECTORIES ...
939. C
940. C 30 CONTINUE
941. C IR = IR+1
942. C
943. C DO 60 I = 1,NTRAJR
944. C I1 = I+1
945. C DO 50 J = I1,NTRAJ
946. C K1 = IORD(I)
947. C K2 = IORD(J)
948. C
949. C ... ACCORDING TO PAST HISTORY ...
950. C
951. C IF(IR.NE.2)KP = IPREC(K1,K2)
952. C IF(KP.EQ.0.AND.IR.EQ.1)GO TO 10
953. C
954. C ... OR ACCORDING TO NOISE LEVEL
955. C
956. C IF(IR.EQ.2)KP = IPREC(K1,K2)
957. C IF(KP.EQ.0)GO TO 40
958. C KM = K1+K2-KP
959. C IORD(I) = KP
960. C IORD(J) = KM
```

```

855. C
856. C UPDATE PAST HISTORY DATA
857. C
858. DO 10 ID = 1,NTRAJ
1 859. VMVT(ID,NTRAJR) = VMCOR(ID)
1 860. ISVT(ID,NTRAJR) = ID
1 861. 10 CONTINUE
1 862. C
1 863. C OBTAIN TRAJECTORY-SELECTION ORDERING
1 864. C
865. CALL ORDER(IP,IM,IU)
866. C
867. C DECIDE WHICH TRAJECTORY IS TO BE BRANCHED
868. C
869. IF(MOD(KGEN,INKPBR).EQ.KPBRO)IM = IP
870. C
871. C PERFORM BRANCHING
872. C
873. DO 20 IC = 1,NDIM
1 874. X(IC,IU) = X(IC,IM)
1 875. 20 CONTINUE
876. H(IU) = H(IM)
877. IE(IU) = IE(IM)
878. DO 30 IT = 1,NTRAJR
1 879. ISVT(IU,IT) = ISVT(IM,IT)
1 880. VMVT(IU,IT) = VMVT(IM,IT)
1 881. 30 CONTINUE
882. EPS(IU) = EPS(IM)
883. DX(IU) = DX(IM)
884. VCOR(IU) = VCOR(IM)
885. DO 40 ID = 1,NTRAJ
1 886. VMCOR(ID) = VMCOR(ID)
1 887. 40 CONTINUE
1 888. C
1 889. C UPDATE PAST-HISTORY-DATA MATRICES
1 890. C
891. CALL COMPAS
892. C
893. C UPDATE SCALING DATA
894. C
895. CALL MOVSCA(IU,IM)
896. C
897. C UPDATE NOISE COEFFICIENT VALUES
898. C
899. IF (EPS(IU).LE.0.00) GO TO 50
900. IF(IFEP.EQ.2)
901. 1 EPS(IU) = EPS(IU)*FACG**((CHAOS(0))-EFAC)
902. IF(IFEP.EQ.1)
903. 1 EPS(IU) = FACG**((DMIN1(DLEPMX,
904. 1 DLOG10(EPS(IU)))+(CHAOS(-1))-EFAC)*DLFACLD)
905. EPS(IU) = DMIN1(EPS(IU),EPSMAX)
906. 50 CONTINUE
907. C
908. C UPDATE MAGNITUDE VOF SPATIAL DISCRETIZATION INCREMENT
909. C
910. DX(IU) = DX(IU)*DFAC**CHAOS(0)
911. RETURN

```

```

806.      1      INT(SNGL(DLOG(DBLE(FLOAT(KGEN))+.500)/DLOG(2.00)))+1
807.      IF(INHP.EQ.2)NKGEN = INT(SNGL(DSQRT(DBLE(FLOAT(KGEN))+.500)))
808.      IF(INHP.EQ.3)NKGEN = KGEN
809.      C
810.      C PERFORM THE REQUIRED NUMBER OF INTEGRATION STEPS
811.      C (FOR ALL TRAJECTORIES)
812.      C
813.      DO 10 IK = 1,NKGEN
814.      C
815.      C PERFORM A SINGLE INTEGRATION STEP
816.      C (FOR ALL TRAJECTORIES)
817.      C
818.      CALL STLP(IK)
819.      10      CONTINUE
820.      C
821.      C MANAGE THE TRAJECTORY BRANCHING PROCESS
822.      C
823.      CALL BRASI
824.      RETURN
825.      END

```

```

826.      SUBROUTINE BRASI
827.      C
828.      C BRASI PERFORMS THE SELECTION PROCESS FOR THE TRAJECTORIES
829.      C BRASI - UPDATES THE DATA ABOUT THE PAST TRAJECTORIES
830.      C - ASKS FOR THE TRAJECTORY-SELECTION ORDERING BY CALLING THE
831.      C SUBROUTINE ORDER
832.      C - DISCARDS ONE OF THE TRAJECTORIES
833.      C - PERFORMS BRANCHING ON ONE OF THE REMAINING TRAJECTORIES
834.      C - MOVES THE DATA OF THE UNPERTURBED CONTINUATION
835.      C TO THE POSITION OF THE PERTURBED CONTINUATION
836.      C - CALLS THE SUBROUTINE COMPAS TO EXAMINE DATA ABOUT PAST
837.      C HISTORY OF THE TRAJECTORIES AND DISCARD IRRILEVANT DATA
838.      C
839.      DOUBLE PRECISION DBAC,DLEPMX,DLFACL
840.      DOUBLE PRECISION EBAC,EPSMAX,FACG
841.      DOUBLE PRECISION CHAOS
842.      DOUBLE PRECISION X,H,DX,VMVT,EPS,VMCOR,VCOR
843.      DOUBLE PRECISION XRMIN,XRMAX,XOPT,FOPT
844.      COMMON /DINCOM/ X(100,20),H(20),DX(20),VMVT(20,19),EPS(20),
845.      1 VMCOR(20),VCOR(20),XRMIN(100),XRMAX(100),XOPT(100),FOPT,
846.      2 IE(20),ISVT(20,19),KGEN,KTIM,NDIM,NTRAJ,NTRAJR,
847.      3 ISEGBR,INKPBR,KPBR,NCF,IFEP,INHPP
848.      C
849.      DATA FACG/10.00/
850.      DATA EFAC/.500/
851.      DATA DFAC/1.03/
852.      DATA EPSMAX/1.015/
853.      DATA DLFACL /0.30102999566398119400/
854.      DATA DLEPMX /25.00/

```

```

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.6M)60 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.  NCF = 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 NMP 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,FPS,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,INHP
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 (INHP.EQ.1)NKGEN=

```



```

1      708.      C
1      709.      C PRINT RESULTS OF OBSERVATION PERIOD
1      710.      C
1      711.      C       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.      C       IF(IR.LT.NPMIN)GO TO 10
1      716.      C       ISTOP = ITOLCH(FMAX,FMIN,TOLREL,TOLABS)
1      717.      C       IF(ISTOP.NE.0)GO TO 30
1      718.      C
1      719.      C 10 CONTINUE
1      720.      C 20 CONTINUE
1      721.      C 30 CONTINUE
1      722.      C CALL NOSCA
1      723.      C
1      724.      C RETURN
1      725.      C END

```

```

726.      SUBROUTINE GENEVA(NX,XMIN,FMIN,FMAX,NCF)
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.      C DOUBLE PRECISION XMIN,FMIN,FMAX
740.      C DOUBLE PRECISION FM
741.      C DOUBLE PRECISION X,H,DX,VMVT,EPS,VMCOR,VCOR
742.      C DOUBLE PRECISION XMIN,XMAX,XOPT,FOPT
743.      C COMMON /DINCOM/ X(800,20),H(20),DX(20),VMVT(20,19),EPS(20),
744.      C 1 VMCOR(20),VCOR(20),XRMIN(100),XRMAX(100),XOPT(100),FOPT,
745.      C 2 IE(20),JSVT(20,19),KGEN,KYIM,NDIM,NTRAJ,NTRAJR,
746.      C 3 ISEGDR,INKPBR,KPBRO,NCF,IFEP,INH
747.      C DIMENSION XMIN(NX)
748.      C
749.      C UPDATE SCALING DATA
750.      C
751.      C DO 10 ID = 1,NTRAJ
1      752.      C CALL SEGSCA(ID)
1      753.      C CALL UPDSCA(NX,X(1,ID))
1      754.      C 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
662.          DO 10 IC = 1, NDJM
663.              X(IC, ID) = XG(IC)
664.              10 CONTINUE
665.              IE(ID) = 0
666.              DO 20 IT = 1, NTRAJR
667.                  ISVT(ID, IT) = 1
668.                  VMVT(ID, IT) = F
669.                  20 CONTINUE
670.                  EPS(ID) = EPS0 * EPSV * (ISEGDR - ID)
671.                  VMCOR(ID) = F
672.                  VCOR(ID) = F
673.                  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, ISTOP)
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.      C0 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.      C DOUBLE PRECISION TOLREL, TOLABS, XMIN, FMIN, FMAX
694.      C DIMENSION XMIN(N)
695.      C DATA IRNF, 77
696.      C
697.      C DO 20 IR = 1, NPMAX
698.      C
699.      C ACTIVATE SCALING
700.      C
701.      C IF (IR.GE.KPASCA.AND.IR.GT.N*IRNF) CALL ACTSCA
702.      C NR = IR
703.      C
704.      C GENERATE AND EVALUATE THE SIMULTANEOUS TRAJECTORY SEGMENTS
705.      C PERIOD
706.      C
707.      C CALL GENEVA(N, XMIN, FMIN, FMAX, NFEV)

```

```

610.      IF(ISEGBR.EQ.0)ISEGBR = (1+NTRAJ)/2
611.      ISEGBR = MIN0(ISEGBR,NTRAJ)
612.      ISEGBR = MAX0(ISEGBR,1)
613.      N2 = ISEGHR
614.      INKPBR = N3
615.      IF(INKPBR.EQ.0)INKPBR = INKPBO
616.      N3 = INKPBR
617.      KPBR0 = N4
618.      IF(KPBR0.EQ.0)KPBR0 = KPBR00
619.      KPBR0 = MOD(KPBR0,INKPBR)
620.      N4 = KPBR0
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
627.          H(ID) = H0
628.          DX(ID) = DX0
629.      20  CONTINUE
630.      C
631.      C INITIALIZE REMAINING VARIABLES
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,KPBR0,NCF,IFEP,INHP
651.      DIMENSION X0(NX)
652.      DATA EPSV/1.00/
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,XO,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 XO,H0,EPS0,DX0,F,XRI,XRA
575.      DOUBLE PRECISION FUNCT
576.      DOUBLE PRECISION X,H,DX,VMVT,EPS,VMCOR,VCOR
577.      DOUBLE PRECISION XMIN,XMAX,XOPT,FOPT
578.      COMMON /DINCOM/ X(100,20),H(20),DX(20),VMVT(20,19),EPS(20),
579.      1      VMCOR(20),VCOR(20),XMIN(100),XMAX(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 XO(NX),XRI(NX),XRA(NX)
583.      DATA NPMAX/100/
584.      DATA NTRAJM/20/
585.      DATA NTRAJO/7/
586.      DATA INKPBO/10/
587.      DATA KPBROO/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.DD.OR.EPS0.LE.0.DD
593.      1      .OR.DX0.LE.0.DD) 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
600.      XMIN(IX) = XRI(IX)
601.      XMAX(IX) = XRA(IX)
602.      10 CONTINUE
603.      CALL NOSCA
604.      NTRAJ = N1
605.      IF(NTRAJ.EQ.0)NTRAJ = NTRAJO
606.      NTRAJ = MIN0(NTRAJ,NTRAUM)
607.      NTRAJ = MAX0(NTRAJ,3)
608.      N1 = NTRAJ
609.      ISEGBR = N2

```

```

1259.          SUBROUTINE NEWH(K,FV,F,H,IE,IEC)
1260.      C
1261.      C NEWH IS CALLED BY THE SUBROUTINE SSTEP TO DECIDE WHETHER TO ACCEPT
1262.      C OR REJECT THE FIRST HALF-STEP, AND TO PROVIDE AN UPDATED VALUE FOR H
1263.      C
1264.          DOUBLE PRECISION FV,F,H
1265.          DOUBLE PRECISION FA,FR,HMAX,HMIN,R
1266.          DIMENSION FR(3),FA(4)
1267.          DATA FR/1.0500,2.00,10.00/
1268.          DATA FA/1.00,1.100,2.00,10.00/
1269.          DATA HMIN/1.0-30/
1270.          DATA HMAX/1.025/
1271.          DATA IECMAX/50/
1272.      C
1273.          IF(FV.LT.F)GO TO 20
1274.      C
1275.      C STEP ACCEPTED, POSSIBLY INCREASE THE STEPLENGTH H
1276.      C
1277.          R = FA(1)
1278.          IF(IE+2.LT.K)R = FA(2)
1279.          IF(IE+3.LT.K)R = FA(3)
1280.          IF(IE.EQ.0.AND.K.GT.1)R = FA(4)
1281.          H = H*R
1282.      10 CONTINUE
1283.          IEC = 0
1284.          GO TO 30
1285.      C
1286.      20 CONTINUE
1287.          IE = IE+1
1288.          IEC = IEC+1
1289.          IF(IEC.GT.IECMAX)GO TO 10
1290.      C
1291.      C STEP REJECTED, DECREASE H
1292.      C
1293.          IC = MIN0(3,IEC)
1294.          R = FR(IC)
1295.          H = H/R
1296.      30 CONTINUE
1297.          H = DMIN1(H,HMAX)
1298.          H = DMAX1(H,HMIN)
1299.      C
1300.          RETURN
1301.          END

```

```

1302.          SUBROUTINE DERFOR(LDIM,X,F,DX,M,DFDX)
1303.      C
1304.      C DERFOR COMPUTED THE FORWARD FINITE-DIFFERENCES DIRECTIONAL
1305.      C DERIVATIVES (CALLING FUNC0 )
1306.      C

```

```

1307.      DOUBLE PRECISION X,F,DX,W,DFDX
1308.      DOUBLE PRECISION DXF,DXFF,DXMAX,FM,S,XX
1309.      DOUBLE PRECISION FUNCTO
1310.      DIMENSION X(NDIM),W(NDIM)
1311.      DIMENSION XX(100)
1312.      DATA DXFF/1.06/,DXB/1.01/
1313.      DATA DXMAX/1.06/
1314.
1315.      C
1316.      10 CONTINUE
1317.      20 CONTINUE
1318.      S = 0.00
1319.      DO 30 IC = 1,NDIM
1320.      XX(IC) = X(IC)+W(IC)*DX
1321.      S = S+(XX(IC)-X(IC))**2
1322.      30 CONTINUE
1323.      IF(S.GT.0.00)GO TO 40
1324.      DX = DX*DXFF
1325.      GO TO 20
1326.      40 CONTINUE
1327.      FM = FUNCTO(NDIM,XX)
1328.      DFDX = (FM-F)/DX
1329.      IF(DX.GT.DXMAX)RETURN
1330.      IF(DABS(DFDX).GT.1.00)RETURN
1331.      IF(DFDX**2.GT.0.00)GO TO 50
1332.      DX = DX*DXF
1333.      GO TO 10
1334.      50 CONTINUE
1335.      C
1336.      RETURN
1337.      END

```

```

1337.      SUBROUTINE DERGEN(NDIM,X,F,DX,W,DFDX)
1338.      C
1339.      C DERFOR COMPUTED THE CENTRAL FINITE-DIFFERENCES DIRECTIONAL
1340.      C DERIVATIVES (CALLING FUNCTO )
1341.      C
1342.      DOUBLE PRECISION X,F,DX,W,DFDX
1343.      DOUBLE PRECISION FM,FR,XX
1344.      DOUBLE PRECISION FUNCTO
1345.      DIMENSION X(NDIM),W(NDIM)
1346.      DIMENSION XX(100)
1347.      C
1348.      DO 10 IC = 1,NDIM
1349.      XX(IC) = X(IC)-W(IC)*DX
1350.      10 CONTINUE
1351.      FR = FUNCTO(NDIM,XX)
1352.      FM = F+DFDX*DX
1353.      DFDX = (FM-FR)/(2.00*DX)
1354.      C
1355.      RETURN

```

1356. END

1357. SUBROUTINE RCLOPT(N,XO,FO)

1358. C
1359. C RCLOPT RECALLS THE CURRENT BEST MINIMUM VALUE FOPT FOUND SO FAR
1360. C FROM ALGORITHM START, AND THE POINT XOPT (OR POSSIBLY ONE OF THE
1361. C POINTS) WHERE FOPT WAS OBTAINED

1362. C
1363. C DOUBLE PRECISION XO,FO
1364. C DOUBLE PRECISION X,H,DX,VMVT,EPS,VMCOR,VCOR
1365. C DOUBLE PRECISION XRMIN,XRMAX,XOPT,FOPT
1366. C COMMON /DINCOM/ X(100,20),H(20),DX(20),VMVT(20,19),EPS(20),
1367. 1 VMCOR(20),VCOR(20),XRMIN(100),XRMAX(100),XOPT(100),FOPT,
1368. 2 IE(20),ISVT(20,19),KGEN,KTIM,NDIM,NTRAJ,NTRAJR,
1369. 3 ISEGBR,INKPBR,KPBR0,NCF,IFEP,INHPT
1370. C DIMENSION XO(N)

1371. C
1372. C DO 10 I = 1,N
1373. 1 XO(I) = XOPT(I)
1374. 10 CONTINUE
1375. C FO = FOPT
1376. C
1377. C RETURN
1378. C END

1379. SUBROUTINE STOOPT(N,XO,FO)

1380. C
1381. C STOOPT STORES THE CURRENT BEST MINIMUM VALUE FOPT FOUND SO FAR
1382. C FROM ALGORITHM START, AND THE POINT XOPT (OR POSSIBLY ONE OF THE
1383. C POINTS) WHERE FOPT WAS OBTAINED

1384. C
1385. C DOUBLE PRECISION XO,FO
1386. C DOUBLE PRECISION X,H,DX,VMVT,EPS,VMCOR,VCOR
1387. C DOUBLE PRECISION XRMIN,XRMAX,XOPT,FOPT
1388. C COMMON /DINCOM/ X(100,20),H(20),DX(20),VMVT(20,19),EPS(20),
1389. 1 VMCOR(20),VCOR(20),XRMIN(100),XRMAX(100),XOPT(100),FOPT,
1390. 2 IE(20),ISVT(20,19),KGEN,KTIM,NDIM,NTRAJ,NTRAJR,
1391. 3 ISEGBR,INKPBR,KPBR0,NCF,IFEP,INHPT
1392. C DIMENSION XO(N)

1393. C
1394. C DO 10 I = 1,N
1395. 1 XOPT(I) = XO(I)
1396. 10 CONTINUE

```

1397.      FOPT = FO
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,VMVT,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,MDIM,NTRAJ,NTRAJR,
1423.      3   ISEGBR,INKPBR,KPBRO,NCF,IFEP,INH
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

```


1446.

END

1447.

SUBROUTINE RANGE (N,XS,XRMIN,XRMAX,R)

1448.

C

1449.

C RANGE IS CALLED BY THE FUNCTION FUNCTD TO TAKE CARE OF THE CASES

1450.

C WHERE THE CURRENT POINT X IS OUTSIDE A GIVEN ADMISSIBLE REGION

1451.

C

1452.

DOUBLE PRECISION XS,XRMIN,XRMAX,R

1453.

DOUBLE PRECISION A,B,C,D,DLRMAX,RMAX,RR,XC

1454.

DIMENSION XS(N),XRMIN(N),XRMAX(N)

1455.

DATA RMAX /1.035/

1456.

DATA DLRMAX/80.590478254791599000/

1457.

C

1458.

R = 0.00

1459.

DO 40 I = 1,N

1

1460.

A = XRMAX(I)

1

1461.

C = XRMIN(I)

1

1462.

XC = XS(I)

1

1463.

IF (XC.LE.A) GO TO 10

1

1464.

B = A+A-C

1

1465.

RR = RMAX

1

1466.

IF (XC.LT.B) RR = DEXP((XC-A)*DLRMAX/(B-A))-1.00

1

1467.

R = R+RR

1

1468.

XS(I) = XRMAX(I)

1

1469.

GO TO 30

1

1470.

10

CONTINUE

1

1471.

IF (XC.GE.C) GO TO 20

1

1472.

D = C+C-A

1

1473.

RR = RMAX

1

1474.

IF (XC.GT.D) RR = DEXP((C-XC)*DLRMAX/(C-D))-1.00

1

1475.

R = R+RR

1

1476.

XS(I) = XRMIN(I)

1

1477.

20

CONTINUE

1

1478.

30

CONTINUE

1

1479.

40

CONTINUE

1

1480.

C

1481.

RETURN

1482.

END

1483.

INTEGER FUNCTION ITOLCH(FMAX,FMIN,TOLREL,TOLABS)

1484.

C

1485.

C ITOLCH DETERMINES WHETHER TWO QUANTITIES ARE TO BE CONSIDERED NU-

1486.

C MERICALLY EQUAL WITH RESPECT TO AN ABSOLUTE (OR RELATIVE) DIFFERENCE

```

1487. C CRITERION, WITHIN GIVEN TOLERANCES
1488. C
1489. C DOUBLE PRECISION FMAX, FMIN, JOLREL, TOLABS
1490. C
1491. C ISTOP = 0
1492. C
1493. C CHECK RELATIVE DIFFERENCE AGAINST TOLREL
1494. C IF(DABS(FMAX-FMIN).LE.TOLREL*(DABS(FMIN)+DABS(FMAX))/2.00)
1495. C 1 ISTOP=ISTOP+1
1496. C
1497. C CHECK ABSOLUTE DIFFERENCE AGAINST TOLABS
1498. C IF(FMAX-FMIN.LE.TOLABS)ISTOP = ISTOP+2
1499. C ITOLCH = ISTOP
1500. C
1501. C RETURN
1502. C END

```

```

1503. SUBROUTINE INISCA(N,ND)
1504. C
1505. C INISCA INITIALIZES THE COMMON AREA /SCALE/ FOR THE SCALING DATA
1506. C
1507. C DOUBLE PRECISION DBST,BIAS,GRAGRA,GRA
1508. C COMMON /SCALE/ DIST(10,10,20),BIAS(10,20),GRAGRA(10,10,20),
1509. C 1 GRA(10,20),NGRA(20),LSCA,IDSCA,NX,NORD
1510. C DATA NXMSCA/10/
1511. C
1512. C LSCA = -1
1513. C IF(N.GT.NXMSCA.OR.N.EQ.1)RETURN
1514. C LSCA = 0
1515. C NX = N
1516. C NORD = ND
1517. C IDSCA = 1
1518. C DO 1 ID = 1,NORD
1519. C DO 2 IX = 1,NX
1520. C DO 3 IV = 1,NX
1521. C DIST(IX,IV,ID) = 0.00
1522. C GRAGRA(IX,IV,ID) = 0.00
1523. C 1 CONTINUE
1524. C DIST(IX,IX,ID) = 1.00
1525. C BIAS(IX,ID) = 0.00
1526. C GRA(IX,ID) = 0.00
1527. C 2 CONTINUE
1528. C 1 NGRA(ID) = 0
1529. C 1 CONTINUE
1530. C
1531. C RETURN
1532. C END

```

```

1533.      SUBROUTINE NOSCA
1534.      C
1535.      C   NOSCA  DEACTIVATES THE SCALING
1536.      C
1537.      DOUBLE PRECISION DIST,BIAS,GRAGRA,GRA
1538.      COMMON /SCALE/ DIST(10,10,20),BIAS(10,20),GRAGRA(10,10,20),
1539.      1   GRA(10,20),NGRA(20),LSCA,IDSCA,NX,NORD
1540.      C
1541.      LSCA = -1
1542.      C
1543.      RETURN
1544.      END

```

```

1545.      SUBROUTINE SEGSCA(ID)
1546.      C
1547.      C   SEGSCA  SELECTS THE TRAJECTORY WHICH MUST BE RESCALED
1548.      C
1549.      DOUBLE PRECISION DIST,BIAS,GRAGRA,GRA
1550.      COMMON /SCALE/ DIST(10,10,20),BIAS(10,20),GRAGRA(10,10,20),
1551.      1   GRA(10,20),NGRA(20),LSCA,IDSCA,NX,NORD
1552.      C
1553.      IDSCA = ID
1554.      C
1555.      RETURN
1556.      END

```

```

1557.      SUBROUTINE VARSCA(N,X)
1558.      C
1559.      C   VARSCA  COMPUTES THE RESCALED VARIABLE AX + B
1560.      C
1561.      DOUBLE PRECISION X
1562.      DOUBLE PRECISION XB
1563.      DOUBLE PRECISION DIST,BIAS,GRAGRA,GRA
1564.      COMMON /SCALE/ DIST(10,10,20),BIAS(10,20),GRAGRA(10,10,20),
1565.      1   GRA(10,20),NGRA(20),LSCA,IDSCA,NX,NORD
1566.      DIMENSION X(N),XB(10)
1567.      C
1568.      IF(LSCA.LE.0)RETURN
1569.      DO 1 I = 1,N
1570.          XB(I) = BIAS(I,IDSCA)
1571.          DO 2 J = 1,N
1572.              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 IRL-CONDITIONING OF
1584.      C F(A+BX) 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(1),40,20,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(0ABS(DFDX).GT.DFDXMA)RETURN
1596.      DO 1 I = 1,N
1597.      DO 2 J = 1,N
1598.      GRAGRA(I,J,IDSCA) = GRAGRA(I,J,IDSCA)+DFDX*W(I)+DFDX*W(J)
1599.      2          CONTINUE
1600.      GRA(I,IDSCA) = GRA(I,IDSCA)+DFDX*W(I)
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(1),30,20,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.      DO 2 J = 1,NX
1630.      DIST(I,J,IU) = DIST(I,J,IM)
1631.      GRAGRA(I,J,IU) = GRAGRA(I,J,IM)
1632.      2 CONTINUE
1633.      BIAS(I,IU) = BIAS(I,IM)
1634.      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 AGRAB,ALA1,ALPHA,AMCOR,BIAS1
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),BIAS1(10),COR(10,10)
1654.      DATA ALPHA /0.3D0/

```

```

1655. C
1656. IF(LSCA.LE.0)RETURN
1657. ID = IDSCA
1658. IF(NGRA(ID).LT.2*NX*NX)GO TO 2
1659. AGMA1 = 1.00/DBLE(DLUAT(NGRA(ID)))
1660. AMCOR = 0.00
1661. DO 3 I = 1,NX
1 1662. DO 4 J = 1,NX
2 1663. COR(I,J) = AGRA1*GRAGRA(I,J,ID)-AGRA1*GRA(I,ID)+AGRA1*GRA(J,ID)
2 1664. AMCOR = DMAX1(AMCOR,DABS(COR(I,J)))
2 1665. CONTINUE
1 1666. CONTINUE
1667. IF(AMCOR.LE.0.00)GO TO J2
1668. DO 1 I = 1,NX
1 1669. DO 11 J = 1,NX
2 1670. COR(I,J) = COR(I,J)/AMCOR
2 1671. CONTINUE
1 1672. CONTINUE
1673. ALA1 = EIGSCA(COR)
1674. CD = ALA1*(1.00+ALPHA)
1675. DO 5 I = 1,NX
1 1676. COR(I,I) = COR(I,I)-CD
1 1677. BIAST(I) = X(I)
1 1678. CONTINUE
1679. CALL VARSCA(NX,BIAST)
1680. SN = 0.00
1681. DO 6 I = 1,NX
1 1682. DO 7 J = 1,NX
2 1683. DISTT(I,J) = 0.00
3 1684. DO 8 K = 1,NX
3 1685. DISTT(I,J) = DISTT(I,J)-DIST(I,K,ID)*COR(K,J)
2 1686. CONTINUE
2 1687. SN = SN+DISTT(I,J)**2
2 1688. CONTINUE
1 1689. CONTINUE
1690. SN = 1.00/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) = 0.00
2 1697. CONTINUE
1 1698. GRA(I,ID) = 0.00
1 1699. CONTINUE
1700. NGRA(ID) = 0
1701. CONTINUE
1702. CONTINUE
1703. C
1704. RETURN
1705. END

```

```

1706.          DOUBLE PRECISION FUNCTION EIGSCA(COR)
1707.          C
1708.          C EIGSCA COMPUTES THE LARGST EIGENVALUE OF A MATRIX USED FOR RES(A-
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,ALAI1,ALA10
1714.          DOUBLE PRECISION PREC,SWM,W,W
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,IBSCA,NX,NORD
1718.          DIMENSION COR(10,10)
1719.          DIMENSION W(10),WW(10)
1720.          DATA PREC /1.0-3/
1721.          DATA NRMIN /10/
1722.          DATA NRMAX /100/
1723.          C
1724.          CALL UNITRV(NX,W)
1725.          ALA1 = 0.00
1726.          DO 1 IR = 1,NRMAX
1727.             ALA10 = ALA1
1728.             SWM = 0.00
1729.             DO 2 IX = 1,NX
1730.                WW(IX) = 0.00
1731.                DO 3 JX = 1,NX
1732.                   WW(IX) = WW(IX)+COR(IX,JX)*W(JX)
1733.                CONTINUE
1734.             SWM = SWM+WW(IX)**2
1735.             CONTINUE
1736.             ALA1 = DSQRT(SWM)
1737.             ALAI1 = 1.00/ALA1
1738.             IF(IR.GE.NRMIN.AND.ALA1*PREC.GT.DABS(ALAI1-ALA10))GO TO 4
1739.             DO 5 IX = 1,NX
1740.                W(IX) = WW(IX)*ALAI1
1741.             CONTINUE
1742.             CONTINUE
1743.             CONTINUE
1744.             EIGSCA = ALA1
1745.          C
1746.          RETURN
1747.          END

```

```

1748.          DOUBLE PRECISION FUNCTION CHAOS(IN17)
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. C DOUBLE PRECISION UNIFRD,PAI,A,B
1763. C
1764. C DATA PAI/3.1415926535897932400/
1765. C
1766. C IF(INIZ.LE.0) GO TO 10
1767. C
1768. C INITIALIZATION.
1769. C
1770. C CHAOS = UNIFRD(INIZ)
1771. C RETURN
1772. C
1773. C 10 CONTINUE
1774. C A = UNIFRD(0)
1775. C IF(INIZ.NE.0) GO TO 20
1776. C B = UNIFRD(0)
1777. C
1778. C GAUSSIAN RANDOM NUMBER BY POLAR METHOD
1779. C
1780. C CHAOS = DSQRT(-2.D0*DLOG(A))*DCOS(PAI*B)
1781. C
1782. C RETURN
1783. C 20 CONTINUE
1784. C
1785. C UNIFORM RANDOM NUMBER IN (0,1)
1786. C
1787. C CHAOS = A
1788. C
1789. C CAUCHY RANDOM NUMBER BY INVERSE TRANSFORMATION
1790. C
1791. C IF(INIZ.EQ.(-1)) CHAOS = DSIN(PAI*A)/DCOS(PAI*A)
1792. C
1793. C UNIFORM RANDOM NUMBER IN (-1,+1)
1794. C
1795. C IF(INIZ.EQ.(-2)) CHAOS = 2.D0*A-1.D0
1796. C
1797. C RETURN
1798. C END

1799. C 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. C     DOUBLE PRECISION A,B,E,X
1807. C     DOUBLE PRECISION X0,P,PD,P1,P2,R1,R2
1808. C     DOUBLE PRECISION FINV
1809. C
1810. C     DIMENSION X(61)
1811. C
1812. C     DATA NREM/61,NRIP/100/
1813. C     DATA A,B,C/-1.500,5.500,-2.000/
1814. C     DATA FINV/3.50-5/
1815. C     DATA IREM/0/
1816. C     DATA PD/3.00/,P1,P2/1.00,3.00/,R1,R2/0.2500,0.7500/
1817. C
1818. C     IF(INIZ.NE.0.OR.IREM.EQ.0) GO TO 10
1819. C
1820. C     IO = IREM
1821. C     X0 = X(IO)
1822. C
1823. C NONLINEARIZATION OF X0 TO AVOID LONG-DISTANCE LINEAR RELATIONSHIPS.
1824. C
1825. C     IF(X0.GE.FINV)X0 = DMOD(1.00/X0,1.00)
1826. C
1827. C UPDATE A COMPONENT OF THE VECTOR X ...
1828. C
1829. C     CALL ALKNUT(NREM,X,IREM)
1830. C
1831. C ... AND FURTHER RANDOMIZE
1832. C
1833. C     UNIFRD = DMOD(X0+X(IO),1.00)
1834. C
1835. C     RETURN
1836. C
1837. C INITIALIZATION OF THE RANDOM NUMBER GENERATOR
1838. C
1839. C 10 CONTINUE
1840. C
1841. C     P = PD-1.00/DBLE(FLOAT(ABS(INIZ))+100.0)
1842. C     DO 20 K = 1,NREM
1843. C
1844. C         P = C+P*(B+P*A)
1845. C
1846. C
1847. C         X(K) = R1+(R2-R1)*(P-P1)/(P2-P1)
1848. C     20 CONTINUE
1849. C
1850. C     IREM = 0
1851. C
1852. C     DO 30 K = 1,NRIP
1853. C
1854. C         CALL ALKNUT(NREM,X,IREM)
1855. C
1856. C     30 CONTINUE
1857. C
1858. C     UNIFRD = X(1)
1859. C

```

```
1860.     RETURN
1861.     END
```

```
1862.     SUBROUTINE ALKNUT(NREM,X,IREM)
1863.     C
1864.     C UPDATES THE COMPONENT IREM OF THE NREM-VECTOR X WITH A RANDOM NUMBER
1865.     C UNIFORMLY DISTRIBUTED IN (0,1) BY MEANS OF THE ALGORITHM
1866.     C OF MITCHELL-MOORE, MODIFIED AS SUGGESTED BY BRENT, QUOTED IN
1867.     C D.E.KNUTH, THE ART OF COMPUTER PROGRAMMING, SECOND EDITION,
1868.     C SECOND VOLUME, SEMINUMERICAL ALGORITHMS, ADDISON-WESLEY
1869.     C PUB. CO., READING (1981), PP. 26-28.
1870.     C
1871.     DOUBLE PRECISION X
1872.     C
1873.     DIMENSION X(NREM)
1874.     C
1875.     DATA N1,N2/24,55/
1876.     C
1877.     IF(IREM.NE.0) GO TO 10
1878.     C
1879.     IREM = NREM
1880.     I1 = NREM-N1
1881.     I2 = NREM-N2
1882.     C
1883.     10 CONTINUE
1884.     C
1885.     X(IREM) = DMOD(X(I1)+X(I2),1.00)
1886.     C
1887.     IREM = 1+MOD(IREM,NREM)
1888.     I1 = 1+MOD(I1,NREM)
1889.     I2 = 1+MOD(I2,NREM)
1890.     C
1891.     RETURN
1892.     END
```

```
1893.     SUBROUTINE GAUSRV(N,W)
1894.     C
1895.     C GENERATES A RANDOM VECTOR SAMPLE FROM AN N-DIMENSIONAL
1896.     C NORMAL DISTRIBUTION
1897.     C
1898.     DOUBLE PRECISION W,X,Y,R
1899.     DOUBLE PRECISION CHADS
1900.     C
```

3.- APPLICAZIONE DELL'ALGORITMO ALL'ANALISI CONFORMAZIONALE

L'algoritmo descritto nel paragrafo 2 è stato concepito per 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

$$E(\underline{\phi}) = E(\phi_1 \dots \phi_m)$$

in cui gli argomenti $\phi_1 \dots \phi_m$ sono gli angoli di torsione liberi.

Il calcolo effettivo di $E(\underline{\phi})$ viene effettuato come segue.

Dati gli angoli $\phi_1 \dots \phi_m$ 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, $t_k = h_0 + h_1 + h_2 + \dots + h_{k-1}$, \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}_0 : 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 $t \rightarrow \infty$, a una densità di probabilità limite

$$p(\underline{x}) = A e^{-\frac{2}{\epsilon} f(\underline{x})}$$

indipendentemente dal punto iniziale \underline{x}_0 . (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 $\epsilon \rightarrow 0$, 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 $\epsilon \rightarrow 0$, 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}_0 e la discretizzazione usata per la (1) ha la forma

$$(2) \quad \underline{x}_{k+1} = \underline{x}_k - h_k \nabla f(\underline{x}_k) + \epsilon(t_k) \sqrt{h_k} \underline{u}_k \quad k = 0, 1, 2, \dots$$

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 $f(\underline{x}) = f(x_1, x_2, \dots, x_n)$ di n variabili reali, seguendo le traiettorie generate da una opportuna discretizzazione numerica dell'equazione differenziale stocastica.

$$(1) \quad d\underline{x} = - \nabla f \, dt + \epsilon \, d\underline{w}$$

[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 $\underline{x} \equiv (x_1, x_2, \dots, x_{3N-6})$ (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#S.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 SIGMAT  
4. C  
5. C DOUBLE PRECISION XD,XMIN,FMIN  
6. C  
7. C DIMENSION XD(2),XMIN(2)  
8. C  
9. C TEST PROBLEM DATA  
10. C  
11. C PROBLEM DIMENSION  
12. C N = 2  
13. C  
14. C INITIAL POINT  
15. C XD(1) = 0.00  
16. C XD(2) = 0.00  
17. C  
18. C SET INPUT PARAMETERS  
19. C NSUC = 3  
20. C IPRINT = 0  
21. C  
22. C CALL DRIVER SUBROUTINE SIGMAT  
23. C CALL SIGMAT(N,XD,NSUC,IPRINT,XMIN,FMIN,NFEV,LOUT)  
24. C  
25. C STOP  
26. C END
```

```
27. C DOUBLE PRECISION FUNCTION FUNCT (N,X)  
28. C  
29. C COMPUTES THE VALUE AT X OF THE SIX-HUMP CAMEL FUNCTION  
30. C  
31. C DOUBLE PRECISION X,XX,VV  
32. C DIMENSION X(N)  
33. C XX = X(1)*X(1)  
34. C VV = X(2)*X(2)  
35. C FUNCT = ((XX/3.00-2.100)*XX+4.00)*XX+X(1)*X(2)  
36. C +4.00*(VV-1.00)*VV  
37. C RETURN  
38. C END
```

```
56. C
57. C STOP
58. C
59. C END
```

```
60. C DOUBLE PRECISION FUNCTION FUNCT(N,X)
61. C
62. C COMPUTES THE FUNCTION VALUES OF TEST-PROBLEM NPROB
63. C BY CALLING THE SUBROUTINE GLOMTF .
64. C
65. C DOUBLE PRECISION X
66. C DOUBLE PRECISION F
67. C DIMENSION X(N)
68. C
69. C COMMON /TUN/ NPROB
70. C
71. C CALL GLOMTF(NPROB,N,X,F)
72. C FUNCT = F
73. C
74. C RETURN
75. C 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. C 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 GLOMIF
12. C
13. C 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. C DIMENSION X0(100),XMIN(100),XMINGL(100),XMAXGL(100)
20. C
21. C 10 CONTINUE
22. C
23. C INPUT PROBLEM NUMBER
24. C WRITE(6,20)
25. C 20 FORMAT(/////41H INPUT PROBLEM NUMBER (1 TO 37, 0 = STOP))
26. C READ(5,30)NPROB
27. C 30 FORMAT(12)
28. C WRITE(6,40)NPROB
29. C 40 FORMAT(///18H PROBLEM NUMBER = ,12////////)
30. C
31. C TERMINATE OR CONTINUE
32. C 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. C 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. C NSUC = 5
42. C
43. C SET IPRINT SO AS TO HAVE A MODERATE OUTPUT
44. C IPRINT = 0
45. C
46. C CALL DRIVER SUBROUTINE SIGMA1
47. C CALL SIGMA1(N,X0,NSUC,IPRINT,XMIN,FMIN,MFEV,IOUT)
48. C
49. C GO TO THE NEXT PROBLEM
50. C GO TO 10
51. C
52. C END OF TEST PROBLEMS
53. C 50 CONTINUE
54. C WRITE(6,60)
55. C 60 FORMAT(/22H END OF TEST PROBLEMS /)
```

```

1901.      DIMENSION W(N)
1902.      C
1903.      C
1904.      C
1905.      DO 20 I = 1,NN
1906.      II = 1+N-I
1907.      C
1908.      10 CONTINUE
1909.      X = CHAOS(-2)
1910.      Y = CHAOS(-2)
1911.      R = X*X+Y*Y
1912.      C
1913.      IF(R.GT.1.00) GO TO 10
1914.      C
1915.      R = DSQRT(-2.00*DLG(R)/R)
1916.      C
1917.      W(I) = X*R
1918.      W(II) = Y*R
1919.      C
1920.      20 CONTINUE
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.      C DOUBLE PRECISION W,WW
1930.      C
1931.      DIMENSION W(N)
1932.      C
1933.      CALL GAUSRV(N,W)
1934.      WW = 0.00
1935.      C
1936.      DO 10 I = 1,N
1937.      WW = WW+W(I)**2
1938.      10 CONTINUE
1939.      WW = 1.00/DSQRT(WW)
1940.      C
1941.      DO 20 I = 1,N
1942.      W(I) = WW*W(I)
1943.      20 CONTINUE
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:

$$E(\phi) = \sum_{i>j} \left(-\frac{A_{ij}}{r_{ij}^6} + \frac{B_{ij}}{r_{ij}^{12}} \right) + \sum_{i=1}^5 \frac{1}{2} K_{\phi_i} (1 + \cos 3\phi_i),$$

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 $\epsilon = 176,2^\circ$, $\zeta = 180,0^\circ$, $\alpha = 122,1^\circ$,

$\beta = -96,7^\circ, \gamma = 55,6^\circ, E = -94,0 \text{ KJmol}^{-1}$), caratterizzato da contatti $H(C3') \dots O6 = 1,75 \text{ \AA}$, $H(C2') \dots O3 = 1,82 \text{ \AA}$, $H(C2') \dots O5' = 2,47 \text{ \AA}$. 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 $\epsilon = -75^\circ, \zeta = 180^\circ, \alpha = 70^\circ, \beta = -110^\circ, \gamma = 55^\circ, E = -91,9 \text{ KJmol}^{-1}$ e la seconda con $\epsilon = -170^\circ, \zeta = 180^\circ, \alpha = 115^\circ, \beta = -100^\circ, \gamma = 55^\circ, E = -90,4 \text{ KJmol}^{-1}$. Il minimo globale ottenuto tramite il metodo qui usato costituisce un guadagno energetico di $3,6 \text{ KJmol}^{-1}$.

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 Transactions 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 Transactions 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.

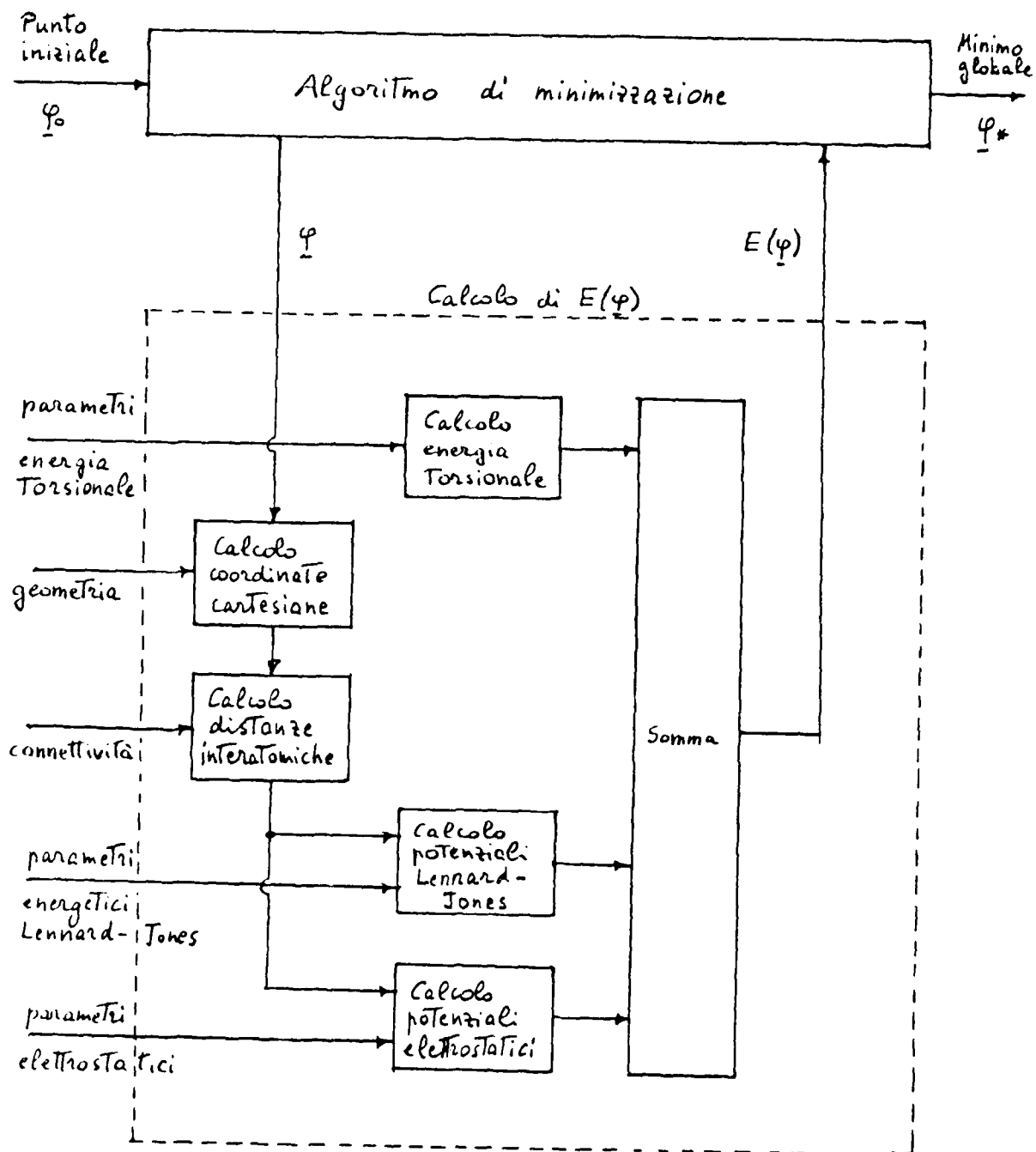


Fig. 1 Diagramma di flusso

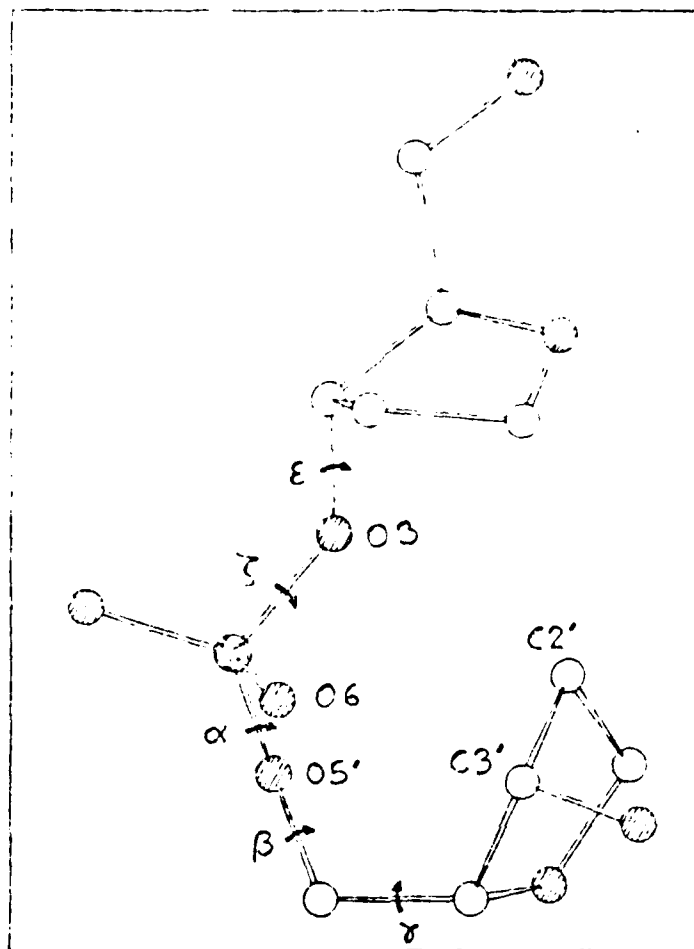


FIGURA 2 - Il frammento C2'(endo) desossi SPS nella conformazione di minima energia trovata con il nuovo algoritmo. I cerchi tratteggiati corrispondono agli atomi di ossigeno e il cerchio pieno all'atomo di fosforo. Per maggiore chiarezza grafica, i 18 atomi di idrogeno sono stati omessi.

END

FILMED

9-85

DTIC