A NEW METHOD FOR GLOBAL OPTIMIZATION BASED ON STOCHASTIC DIFFERENTIAL EQUATIONS (ITALY) CAMERINO UNIV (ITALY) MATHEMATICS INST F ALUFFI-PENTINI ET AL.

UNCLASSIFIED DEC 84 DAJA37-81-C-0740
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
A new approach is presented to the problem of finding a global (i.e., absolute) minimizing 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.
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


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).


A6 The FORTRAN package SIGMA.

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 \( x_1, x_2, \ldots, x_N \), i.e. the point \( \mathbf{x}^* = (x_1^*, \ldots, x_N^*) \) in the \( N \)-dimensional real Euclidean space \( \mathbb{R}^N \) such that \( f \) attains at \( \mathbf{x}^* \) a global (or "absolute") minimum, defined by

\[
(1) \quad f(x^*) \leq f(x) \quad \text{for all } \mathbf{x} = (x_1, \ldots, x_N) \in \mathbb{R}^N
\]

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

\[
(2) \quad \lim_{\|\mathbf{x}\|_2 \to \infty} f(\mathbf{x}) = +\infty
\]

in such a way that

\[
(3) \quad \int_{\mathbb{R}^N} \exp(-2f(x)/\epsilon^2) \, d\mathbf{x} < +\infty
\]

for all real \( \epsilon \neq 0 \).

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 \( g(x) = 0 \), where \( g: \mathbb{R}^N \to \mathbb{R}^N \) can be formulated as a global optimization problem considering the function

\[
F(\mathbf{x}) = \|g(\mathbf{x})\|_2^2 \quad \text{where } \| \cdot \|_2 \quad \text{is the Euclidean norm in } \mathbb{R}^N.
\]

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

\[
\frac{dx}{dt} = -\nabla f(x)
\]

where \( \nabla f \) is the gradient of the function \( f \) to be minimized.

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

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

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

\[
d\xi = -\nabla f(\xi) \, dt + \epsilon \, dw
\]

where \( \nabla f \) is the gradient of the function \( f \) to be minimized, \( w(t) \) is a standard \( N \)-dimensional Wiener process ("Brownian motion"), and \( \epsilon \) 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 \( \epsilon dw \), where

\[
\epsilon = \left( \frac{2kT}{m} \right)^{\frac{1}{2}}
\]

\( T \) is the absolute temperature, \( k \) the Boltzmann constant, \( m \) a suitable mass coefficient, and \( f \) is the potential energy.

The use of the above equation is suggested by the behaviour, for constant \( \epsilon \), 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
\( p_\varepsilon(x,t) \) of the (random) value at time \( t \) of the solution process tends, as \( t \to \infty \) (if condition (3) holds), to a limit "equilibrium" density

\[
p_\varepsilon(x) = A_\varepsilon e^{-(2/\varepsilon^4)f(x)}
\]

where \( A_\varepsilon \) is a normalization constant.

The equilibrium density is independent of the starting point \( x_0 \) and is peaked at the global minimizers of \( f \), with narrower peaks if \( \varepsilon \) is smaller.

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

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

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

\[
d\xi = -\nabla f(\xi)dt + \varepsilon(t)dw
\]

with initial condition

\[
\xi(0) = x_0
\]

where

\[
\lim_{t \to \infty} \varepsilon(t) = 0.
\]

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

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

Several mathematical questions related to the solutions of eqs. (5) and (7), such as the way in which \( p_\varepsilon(x,t) \) approaches \( p_\varepsilon(x) \) for a class of one-dimensional systems, or the rate at which \( \varepsilon(t) \) should go to 0 in eq. (9), are considered in Appen-
dices 1 and 2.

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

3. The algorithm SIGMA

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

\[
d\xi = -vf(\xi)dt + \varepsilon(t)dw
\]

with initial condition:

\[
\xi(0) = x_0
\]

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

\[
\xi_{k+1} = \xi_k - h_k \frac{\partial^2 f(\xi_k)}{\partial x^2} + \varepsilon(t_k) \left( h_k \text{U}_k \right)^{\frac{1}{2}}, \quad k = 0, 1, 2, ...
\]

\[
\xi_0 = x_0
\]

where \( h_k \) is the time integration steplength, \( \frac{1}{N} \sum_{k=1}^{N} \xi_k \) is computed as a finite-differences approximation to the directional derivative of \( f \) in a randomly chosen direction, and \( \text{U}_k \) is a random sample from an N-dimensional standard gaussian distribution.

We consider the simultaneous evolution of a number \( N_{\text{TRAJ}} \) of trajectories during an "observation period" having the duration of a given number \( N_{\text{HP}} \) of the time integration steps, and within which the noise coefficient \( \varepsilon(t) \) of each trajectory is kept at a constant value \( \varepsilon_p \), while the steplength \( h_k \) and the spatial in-
Clearly \( p_{\omega}^{(n)} \) is the probability density of a random variable \( \hat{\omega} \)
so that \( \hat{\omega}(t) \sim \omega_{\omega} \) in law when \( t \to \infty \). Let us remark that \( p_{\omega}^{(n)} \)
does not depend on the initial condition \( x_0 \).

We want to study the behaviour of \( p_{\omega}^{(n)} \) as \( \omega \to 0 \) and the rate
of approach of \( p_{\omega}^{(n)} \) to \( p_{\omega}^{(0)} \) as \( t \to \infty \). We will consider for the sake
of simplicity only the one-dimensional case when \( f \) is as in Fig. 1,
i.e., with three extrema at the points \( x_0 = x_1 < x_2 \), decreasing in
\((x_0, x_1)\) and \((x_1, x_2)\) and increasing in \((x_2, x_3)\) and \((x_3, +\infty)\), with
\( f(x) = 0 \) as \( x \to -\infty \) in such a way to satisfy (11) for all \( n \).

We have

\[
\frac{df}{dx}(x_1) = \frac{df}{dx}(x_2) = \frac{df}{dx}(x_3) = 0.
\]

Define the following notation

\[
\begin{align*}
f_+ &= f(x_2), & c_+ &= \frac{df}{dx}(x_2) \\
f_- &= f(x_1), & c_- &= \frac{df}{dx}(x_1) \\
f &= f(x_0), & c &= -\frac{df}{dx}(x_0) \\
f_0 &= f_+ - f_-, & f_{f+} &= f_+ - f_+ > 0
\end{align*}
\]

It is easy to prove the following result.

**Proposition 2.1.** Let \( f \) be as above and let \( c_-, c_+ \), \( c \) be greater than
zero, then

\[
f = f_+ - f_- \quad \text{and} \quad f_{f+} = f_+ - f_+ > 0
\]

then

\[
f(x) = -\frac{f_+ f_0}{f_{f+}} (x-x_1) + f_+ \quad \text{for} \quad x \in \mathbb{R}
\]
Let $\xi^{t_0}(t)$ be the stochastic process solution of (5), (6); for any Borel set $A \subset \mathbb{R}^n$ we define

$$P^t(0,x_0,t,A) = \mathbb{P}\{\xi^{t_0}(t) \in A\}$$

where $\mathbb{P}(\cdot)$ is the probability of $\{\cdot\}$, and $P^t(0,x_0,t,A)$ is the transition probability of $\xi^{t_0}(t)$. Under regularity assumptions for $f$ we have

$$P^t(0,x_0,t,A) = \int_A p^t(0,x_0,t,x)dx$$

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

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

with

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

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

Let $A_{t_0}$ be defined by

$$\frac{1}{A_{t_0}} = \int_{\mathbb{R}^n} e^{-\epsilon^2 f(x)/t_0^2} dx < \infty$$

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

$$p^t(0,x_0,t,x) = A_{t_0} e^{-\epsilon^2 f(x)/t_0^2}$$
2. Method

Let us consider the Cauchy problem

\[ d\xi = -\nabla f(\xi) \, dt + \varepsilon(t) \, dw \]  \tag{3} \\
\xi(0) = x_0 \tag{4} 

for the (Ito) stochastic differential equation (3), where \( f: \mathbb{R}^n \to \mathbb{R} \) is the function to be (globally) minimized, \( \nabla f \) 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 \( \varepsilon \) are sufficiently well-behaved so that our statements are meaningful; in particular we assume that

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

and

\[ \int_{\mathbb{R}^n} e^{-\nabla f(x) \cdot x} \, dx < \infty \quad \forall a \in \mathbb{R}: (0) \]

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 \( \varepsilon(t) = \varepsilon \) is a constant; that is

\[ d\tilde{\xi} = -\nabla f(\tilde{\xi}) \, dt + \varepsilon \, dw(t) \]  \tag{5} \\
\tilde{\xi}(0) = x_0 \.
minimizers of $f$, with narrower peaks if the constant $\varepsilon_0$ is smaller.

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

In Section 2, we describe our method; in Section 3, we consider the numerical integration problem; and in Section 4, we present the results of numerical experiments on several test problems.
that is, it depends on the behavior of \( f \) on each point of \( \mathbb{R}^n \), and the
methods that follow a trajectory of a system of ordinary differential equa-
tions 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 equa-
tions. However, the situation is different if we consider a suitable sto-
chastic perturbation of a system of ordinary differential equations as we
now describe.

Let us consider the (Itô) stochastic differential equation

\[ \frac{d\xi}{\xi} = -f(\xi)dt + \sigma dw \]

where \( f \) is the gradient of \( f \) and \( w(t) \) is a standard \( n \)-dimensional
Wiener process. When \( \sigma = \sigma \) 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 chem-
tists to study physical phenomena such as atomic migration in crystals or chemi-
cal reactions. In these applications \( \sigma = \sqrt{\frac{2kT}{m}} \) 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 \( \sigma dw \).

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

Let $\mathbb{R}^n$ be the $n$-dimensional real Euclidean space $x = (x_1, x_2, \ldots, x_n)^T \in \mathbb{R}^n$ and let $f: \mathbb{R}^n \rightarrow \mathbb{R}$ 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, \( x = (x_1, x_2, \ldots, x_n)^T \in \mathbb{R}^n \) 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-PENTINI$^3$, V. PARISI$^4$, AND F. ZIRILLI$^5$

Communicated by R.A. Tapia

---

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

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

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

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

$^5$Professor, Istituto di Matematica, Università di Salerno, Salerno, Italy.
APPENDIX A1

Global optimization and stochastic differential equations

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

<table>
<thead>
<tr>
<th>( N_{SUC} )</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Totals (1)</td>
<td>26</td>
<td>32</td>
<td>34</td>
<td>35</td>
<td>35</td>
<td>32</td>
<td>34</td>
<td>34</td>
<td>34</td>
<td>35</td>
</tr>
<tr>
<td>(2)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>(3)</td>
<td>11</td>
<td>5</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>5</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>(4)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

(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 sub-
mitted to the ACM Transaction on Mathematical Software (see Ap-
pendix 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 pro-
blems 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 cor-
responding dynamic range of about $10^{\pm 10}$ 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 effective-
ness, dependability and robustness - in the form of total num-
ers of correctly claimed successes, correctly claimed failures,
incorrect success or failure claims and total number of over-
flows - for the two machines and for different values of $N_{\text{SUCC}}$ (sect. 3).

The SIGMA package seems to perform quite well on the propo-
sed test problems.

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

Table 1 shows that from the point of view of the effective-
ness 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 \( \Delta x_k \) for computing \( \gamma_k \) 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 \( \varepsilon_p \) and \( \Delta x_k \), and is considered as having the same "past history" of the first.

The number \( N_{\text{TRAJ}} \) 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 \( N_{\text{PMAX}} \) of observation periods has been reached, or if all the final values of \( f(x) \) (except for the discarded trajectory) are equal (within numerical tolerances, and possibly at different points \( x \)) to their minimum value \( f_{\text{TFMIN}} \) ("uniform stop" at the level \( f_{\text{TFMIN}} \)). In the former case the trial is considered unsuccessful, while in the latter case a comparison is made between the common final function value \( f_{\text{TFMIN}} \) and the current best minimum function value \( f_{\text{OPT}} \) 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 \( f_{\text{OPT}} \).

The trials are repeated with different operating conditions (initial point \( x_0 \), maximum trial length \( N_{\text{PMAX}} \), seed of the noise generator, policy for selecting the starting value for \( \varepsilon_p \) in the second continuation trajectory after branching, and trial-start values for \( \varepsilon_p \) ) and the complete algorithm is stopped - at the end of a trial - if a given number \( N_{\text{SUC}} \) of uniform stops at the current \( f_{\text{OPT}} \) level has been obtained, or if a given maximum number \( N_{\text{TRIAL}} \) of trials has been reached: success of the algorithm is claimed if at least one uniform stop occurred at the final value of \( f_{\text{OPT}} \).

A detailed description of the algorithm is given in Appendix A5.
\[
\lim_{\varepsilon_0 \to 0} p_{\varepsilon_0}^0(0,x_0,x) = \delta(x - x_-) \tag{13}
\]

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

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

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

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

Remark 2.1. Proposition 2.1 shows that as \( \varepsilon_0 \to 0 \), the asymptotic probability density approaches a Dirac delta function concentrated on the global minimizer when there is a unique global minimizer (\( \Delta f_- > \Delta f_+ \)), or approaches a linear combination of Dirac delta functions concentrated on the global minimizers (\( \Delta f_- = \Delta f_+ \)). The coefficients of the linear combination depend on the curvature of \( f \) at the global minimizers. These statements have a clear meaning in terms of \( p_{\varepsilon_0}^0 \). 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 \( p_{\varepsilon_0}^0 \) to \( p_{\varepsilon_0}^0 \) as \( t \to \infty \) by looking at the eigenvalues of the Fokker-Planck operator

\[
l_{\varepsilon_0}(\cdot) = \frac{\varepsilon_0^2}{2} \frac{\partial^2(\cdot)}{\partial x^2} + \frac{\partial}{\partial x} \left( \frac{df}{dx} \cdot \right)
\]

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

\[
\lambda_1(\varepsilon_0) \simeq -\frac{\sqrt{c_- c_0}}{2\pi} e^{-\frac{2}{\varepsilon_0^2} \Delta f_+} \tag{15}
\]
So that roughly speaking we can imagine:

\[ p_\varepsilon^@ (0, x, t, x) = p_0^\varepsilon + \exp \left\{ \int_0^t \lambda_1 (\varepsilon_s) ds \right\} \hat{p} \]  

(16)

where \( \hat{p} \) is an eigenfunction corresponding to \( \lambda_1 \).

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

Remark 2.2. Since \( \lambda_1 (\varepsilon_s) \to 0 \) as \( \varepsilon_0 \to 0 \) from (16) we see that the rate of approach to \( p_\varepsilon^\varepsilon \) became slower when \( \varepsilon_0 \) became smaller. On the other hand from (12) we see that \( p_\varepsilon^\varepsilon \) becomes more and more concentrated around the global optimizers as \( \varepsilon_0 \) 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 \( P(0, x_0, t, \xi) \) 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:

\[ \frac{\partial p}{\partial t} = \frac{-f(x)}{\partial x} - \Delta p + \text{div} (\sigma f p) \]  

(17)

\[ \lim_{t \to 0} p(0, x_1, t, x) = \delta(x - x_0) \]  

(18)

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

\[ \lim_{t \to \infty} p(0, x_0, t, x) = \frac{m}{\sum_{i=1}^m r_i} (x - x_i^*) \]  

(19)

where \( r_i \) are positive constants such that \( \sum_{i=1}^m r_i = 1 \) and \( x_i^* \).
i = 1, 2, 3, ..., m are the global minimizers of f.

The previous analysis of the corresponding problem with \( \epsilon(t) = \epsilon_0 \) suggests that in order to have (19) we need

\[
\lim_{t \to \infty} \epsilon(t) = 0 \quad (20)
\]

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

\[
\int_0^\infty e^{-\frac{2}{\epsilon^2(t)}} \Delta f_+ \, dt = \infty \quad (21)
\]

where \( \Delta f_+ \) is the highest barrier to the global minimizers. We note that in order to satisfy (21) \( \epsilon(t) \) 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 \( \zeta(t) \) when \( t \to \infty \). We want to consider here the problem of how to compute numerically these paths keeping in mind that we are not really interested in the paths, but only in their asymptotic values.

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

Let \( \Delta t_k = 0 \), \( t_k = \frac{k}{2} \Delta t \) \((t_0 = 0), k = 0, 1, \ldots \); we discretize (3), (4) using the Euler-Cauchy method, that is \( \zeta(t_k) \) is approximated by \( \zeta_k \) solution of the following finite difference equations:

\[
\begin{align*}
\zeta_{k+1} - \zeta_k &= -\Delta t_k \left( f(\zeta_k) + \zeta(t_k)(w_{k+1} - w_k) \right) \\
\zeta_0 &= x_1,
\end{align*}
\]

(22)

Since for stability reasons \( \Delta t_k \) will be chosen rather small and since condition (21) implies that \( \zeta(t) \) 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 \( \langle v, r \rangle \) along \( r \) is such that

\[
\text{proj}(\langle v, r \rangle r) = v
\]
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. functions evaluations) we may substitute to $\nabla f(\xi_k)$ in eq. (22) the expression

$$n < \nabla f(\xi_k), r > r$$

(24)

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

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

We have considered this alternative strategy:

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

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

The new path has a different value of $\varepsilon(t) = \text{constant}$; $\varepsilon(t)$ is usually decreased, occasionally it can be increased if the paths
are stuck in a local minimizer as detected by looking at the previously computed values of \( f \).

(iii) repeat step (ii).
4. **Test Problems and Numerical Experience**

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

We shall make use of the penalization function

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

The test problems are:

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

- \( x = -3, \quad f(x) = 7, \)
- \( x = 0, \quad f(x) = 250, \)
- \( x = 3, \quad f(x) = 7. \)

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

**Problem 2. Penalized Shubert Function.** Let \( g_1(x) = \sum_{i=1}^{5} \cos((i+1)x+1); \) 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 \( \{x | |x| < 10\} \) and three of them are global ones and they are located at:
Problem 3. Two-dimensional Penalized Shubert Function. Let

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

\[ + u(x, 10, 100, 2) + u(y, 10, 100, 2) \]

The function \( f \) has 50 minima, (18 of them are global minima) in the region \( -10 \leq x, y \leq 10 \).

Problem 3. Two-dimensional Penalized Shubert Function \( z = 0.5 \).

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

\[ + \{(x+1.42515)^2 + (y+0.80032)^2\} \]

\[ + u(x, 10, 100, 2) + u(y, 10, 100, 2) \]

where \( z = 0.5 \) and \((-1.42515, -0.80032)\) is a point where the function \( f \) with \( z = 0 \) has a global minimizer.

This function has roughly the same behaviour of the function considered in problem 3 but has a unique global minimizer at \((-1.42515, -0.80032)\) where the function \( f \) is equal to \(-186.7309\).

Problem 3. Two-dimensional Penalized Shubert Function \( z = 1 \). The function \( f \) is the one given in problem 3 with \( z = 1 \).

Problem 5. Camel Function. Let \( f \) be given by

\[ f(x, y) = (1 - 2.1x^2 + \frac{x^4}{3}) x^2 + xy + (-4 + 4y^2)y^2 \]

The function has 5 minima, two of them are global minima and are located at \((-1.25, 0), (-0.938, 0), (-0.025, 0)\).
Problems 7-9 are obtained from the following formula:

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

(25)

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

\( k_2 = 10, A_2 = 1. \)

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

\[
x_i = 1 \quad i = 1, 2, \ldots, n.
\]

We penalize the function (25) as follows:

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

(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:

\[
g_3(x) = \frac{\pi}{n} \{k_3 \sin^2 \pi x_i + \sum_{i=1}^{n-1} (x_{i-1} - A_3)^2 + (1 + k_3 \sin^2 \pi x_{i+1}) + (x_i - A_3)^2\}
\]

(27)

where \( k_1 = 10, A_3 = 1 \) and \( x = (x_1, x_2, \ldots, x_n)^T. \)

In the region \( \Omega = \{x \in \mathbb{R}^n | -10 \leq x_i \leq 10 \ i = 1, 2, \ldots, n\} \) the function (27) has roughly \( 10^n \) local minimizers and a unique global minimizer at \( x_i = 1 \)

\( i = 1, 2, \ldots, n. \) We penalize the function (27) as follows:
\[ f(x) = g_n(x) + \sum_{i=1}^{n} u(x_i, \theta, \alpha, \beta) \]  \hspace{1cm} (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 13-16 are obtained from the following formulas:

\[ g_n(x) = k_i \sin^{-1} k_i x_i + \sum_{i=1}^{n-1} (x_i - A_i)^2 (1 + k_i \sin^{-1} k_i x_i) \]

\[ + (x_n - A_i)^2 (1 + k_i \sin^{-1} k_i x_i) \]  \hspace{1cm} (29)

where \( k_i = 0.1, \ k_i = 1, \ A_i = 1, \ k_i = 3, \ k_i = 2 \).

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

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

\[ f(x) = g_n(x) + \sum_{i=1}^{n} u(x_i, \theta, \alpha, \beta) \]  \hspace{1cm} (30)

or

\[ f(x) = g_n(x) + \sum_{i=1}^{n} u(x_i, \theta, \alpha, \beta) \]  \hspace{1cm} (31)

**Problem 15.** The function \( f(x) \) is given by (30) with \( n = 2 \).

**Problem 16.** The function \( f(x) \) is given by (30) with \( n = 5 \).
Problem 15. The function \( f(x) \) is given by (30) with \( n = 4 \).

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

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

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

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

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

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

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

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

Problem 23. Let \( f(x) = \left( \sum_{i=1}^{5} x_i^2 \right)^{\frac{1}{2}} \), where \( x = (x_1, \ldots, x_5)^T \), the function \( f(x) \) has a unique minimizer at \( x = 0 \) where the function is not differentiable, moreover the hessian of \( f(x) \) is not defined at \( x = 0 \) and is not positive definite in a neighborhood of \( x = 0 \).
The remaining problem 24 has been suggested by S. Wolff, Ref. 8.

**Problem 24.** Let

\[ f(x,y) = -f(x,y) + u(x,10^4,100,2) + u(y,10^4,100,2) \]

where

\[ F(x,y) = \sum_{i=1}^{n} \left[ \frac{x_i - x}{y} \right]^{1-\delta} i \left[ 1 - \frac{(\frac{x_i - x}{y})^{3-\delta}}{} \right] \]

The data points \( x_i \) are given by:

\[
\begin{array}{cccccccc}
  x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 & x_8 \\
  1219 & 1571 & 1377 & 1144 & 1201 & 1225 & 1244 & 1264 \\
  0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\
\end{array}
\]

\[
\begin{array}{cccccccc}
  y_1 & y_2 & y_3 & y_4 & y_5 & y_6 & y_7 & y_8 \\
  1253 & 1324 & 1328 & 1351 & 1356 & 1370 & 1390 & 1390 \\
  1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
\end{array}
\]

and \( \psi(x) = \int_0^\infty e^{-t^2/2} \frac{dt}{\sqrt{2\pi}} \).

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

The numerical results obtained are shown in Table 1.
<table>
<thead>
<tr>
<th>Problem</th>
<th>NFI</th>
<th>Whether a global minimizer has been found</th>
<th>NF2</th>
<th>Whether a global minimizer has been found</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3,184</td>
<td>Yes</td>
<td>7,168</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>26,893</td>
<td>Yes</td>
<td>77,699</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>3,218</td>
<td>No</td>
<td>241,215</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>8,755</td>
<td>Yes</td>
<td>70,894</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>97,761</td>
<td>Yes</td>
<td>185,819</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>5,393</td>
<td>Yes</td>
<td>10,822</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>84,782</td>
<td>Yes</td>
<td>150,549</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>19,041</td>
<td>Yes</td>
<td>72,851</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>18,942</td>
<td>Yes</td>
<td>49,690</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>18,433</td>
<td>Yes</td>
<td>72,226</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>4,322</td>
<td>No</td>
<td>150,061</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>49,701</td>
<td>Yes</td>
<td>98,985</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>9,492</td>
<td>Yes</td>
<td>23,770</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>19,114</td>
<td>Yes</td>
<td>66,010</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>35,139</td>
<td>Yes</td>
<td>122,166</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>53,398</td>
<td>Yes</td>
<td>66,365</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>15,534</td>
<td>Yes</td>
<td>98,974</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>16,542</td>
<td>Yes</td>
<td>109,886</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>6,751</td>
<td>Yes</td>
<td>10,487</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>3,402</td>
<td>Yes</td>
<td>12,249</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>10,286</td>
<td>Yes</td>
<td>19,940</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>4,791</td>
<td>Yes</td>
<td>7,390</td>
<td>Yes</td>
<td>n=m=1</td>
</tr>
<tr>
<td>22</td>
<td>3,037</td>
<td>Yes</td>
<td>4,663</td>
<td>Yes</td>
<td>n=m=2</td>
</tr>
<tr>
<td>22</td>
<td>5,028</td>
<td>Yes</td>
<td>8,235</td>
<td>Yes</td>
<td>n=m=3</td>
</tr>
<tr>
<td>22</td>
<td>14,710</td>
<td>Yes</td>
<td>27,859</td>
<td>Yes</td>
<td>n=m=4</td>
</tr>
<tr>
<td>22</td>
<td>51,285</td>
<td>Yes</td>
<td>74,194</td>
<td>Yes</td>
<td>n=m=5</td>
</tr>
</tbody>
</table>
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 values for (t) etc. ...) are fixed once and for all during the runs.

The initial point \( x_i \) has been chosen as follows:

- \( x_{10} = 0 \) for Problems 1-18
- \( x_{19} = 1/50 \) for Problem 19
- \( x_{20} = (1, 0) \) for Problem 20
- \( x_{21} = (-5, 0) \) for Problem 21
- \( x_{22} = (0, 1) \) for Problem 22
- \( x_{23} = (10^1, 10^2, \ldots, 10^7) \) for Problem 23
- \( x_{24} = (-1.250, -10000) \) for Problem 24.

For Problems 19-22 and 24 the initial point \( x_i \) has been chosen close to a local minimizer.

The condition number of the tangent at the solution of Problem 22 increased with \( n \), i.e., the solution of the solution of Problem 23 is not defined; the solution at the solution of Problem 24 is ill-conditioned; the remaining problems were well-conditioned. The table of 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:

<table>
<thead>
<tr>
<th>( y )</th>
<th>( \sqrt{\frac{4\alpha^2 - 4\alpha^4 + 9\epsilon^2}{3\epsilon^2}} )</th>
<th>( \frac{\sqrt{6}}{3\epsilon} )</th>
<th>( \frac{\sqrt{2}}{\epsilon} )</th>
<th>( \sqrt{\frac{4\alpha^2 + \sqrt{4\alpha^4 + 9\epsilon^2}}{3\epsilon^2}} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( V_\epsilon(y) )</td>
<td>( 2\epsilon )</td>
<td>( \frac{2}{27\epsilon^2} \left( 8\gamma^6 - 27\alpha^2 \epsilon^2 \right) + (4\alpha^4 + 9\epsilon^2)^{3/2} )</td>
<td>( \frac{32\alpha^6}{27\epsilon^2} )</td>
<td>( -4\alpha^2 )</td>
</tr>
<tr>
<td>( V_\epsilon'(y) )</td>
<td>0</td>
<td>0</td>
<td>( -2\epsilon \sqrt{6} )</td>
<td>( -6\epsilon \sqrt{2} )</td>
</tr>
<tr>
<td>( V_\epsilon''(y) )</td>
<td>( 8\alpha^6 - 6\epsilon^2 )</td>
<td>( \frac{8}{3} (4\alpha^4 + 9\epsilon^2) )</td>
<td>( \frac{32\alpha^6}{3} ) - ( 6\epsilon^2 )</td>
<td>( 32\alpha^6 - 6\epsilon^2 )</td>
</tr>
</tbody>
</table>

Table 1

(\text{where}'\text{' means differentiation})

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

(iv) \( V_\epsilon(y) \) is given by Fig. 3.
\[ U_\varepsilon(y) = \frac{1}{2} \left( \frac{d^2 f_\varepsilon}{dy^2} \right)^2 - \frac{d^2 f_\varepsilon}{dy^2} \]

\[ = 4y^2 \left( 2ae^2y^2 + \frac{3be}{\sqrt{2}} y + 2c \right)^2 - \frac{1}{2} \left( 6ae^2y^2 + \frac{6be}{\sqrt{2}} y + 2c \right) = \]

\[ = a^2 \varepsilon^4 y^6 + \frac{3ab}{\sqrt{2}} \varepsilon^3 y^5 + \left( \frac{b^2}{9} + \frac{2ac}{2} \right) \varepsilon^2 y^4 + \frac{3bc}{\sqrt{2}} \varepsilon y^3 + \]

\[ + (c^2 - 3ae^2) y^2 - \frac{3be}{\sqrt{2}} y - c \]

In order to understand intuitively the behavior as \( \varepsilon \to 0 \) of the spectrum of \( H_\varepsilon \) when the potential \( W_\varepsilon(y) \) is given by \( V_\varepsilon \) or \( U_\varepsilon \), let us analyze the behavior of \( V_\varepsilon \) and \( U_\varepsilon \) when \( \varepsilon \to 0 \).

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

(i) the equation

\[ \frac{dV_\varepsilon}{dy}(y) = 0 \]

has five real roots

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

that is, \( V_\varepsilon \) has a local minimizer at \( y = 0 \), two global minimizers

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

and two local maximizers at
Fig. 1

Fig. 2
In this paper we will consider the case when $f(x)$ is given by

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

or by

$$f_2(x) = x^4 (ax^2 + bx + c), \quad x \in \mathbb{R}$$

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

$$a > 0$$

$$b^2 - 4ac < 0$$

$$9b^2 - 32ac > 0$$

$$b < 0$$

Since the spectrum of $H_0$ 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 $f_2(x) > 0 \ \forall x \in \mathbb{R}$, with $f_2(x) = 0 \iff x = 0$, (2.17) implies that $f'_2(x) = 0$ has three real roots $0, x_1, x_2$ and that $f''_2(x) = 0$ has two real roots, that is $x_1$ is a maximizer of $f_2$ and $x_2$ is a minimizer of $f_2(x)$; finally (2.18) implies that $0 < x_1 < x_2$ (Fig. 2).

A straightforward computation gives:

$$V_0(y) = -\frac{df_2}{dy} \frac{df_2}{dy} - \frac{d^2f_2}{dy^2} =$$

$$= x^4y^4 - 4x^2y^2 + (4x^2 - 3^2)y^2 + 2x^2$$
Let us note that $H_\varepsilon$ is a Schrödinger Hamiltonian. It is easy to verify that

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

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

$$u_0(x) = c_\varepsilon e^{-2k^2 f(x)} \quad x \in \mathbb{R}$$

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

$$\int_{-\infty}^{+\infty} e^{-2k^2 f(x)} \, dx < \infty \quad \forall \varepsilon \neq 0$$

and we will choose

$$c_\varepsilon = \left[ \int_{-\infty}^{+\infty} e^{-2k^2 f(x)} \, dx \right]^{-1}$$

so that

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

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 $H_\varepsilon$ in $L^2(\mathbb{R})$. 
§2. From the Fokker-Planck equation to the Schrödinger equation.

Let us consider the eigenvalue problem

\[ L_u u = \lambda u, \quad \lambda \in \mathbb{C}, \quad x \in \mathbb{R} \tag{2.1} \]

where \( L_u \) is given by (1.4).

Let us consider the change of variables

\[ y = \frac{x}{\sqrt{\varepsilon}} \tag{2.2} \]

\[ v(y) = c^{-\frac{1}{2}} e^{-\frac{f_u(y)}{2}} u\left( \frac{y}{\sqrt{\varepsilon}} \right) \tag{2.3} \]

where \( c \) is a normalization constant and

\[ f_u(y) = \frac{2}{\varepsilon^2} f\left( \frac{y}{\sqrt{\varepsilon}} \right) \tag{2.4} \]

The eigenvalue problem (2.1) becomes

\[ H_v v = -\lambda v, \quad \lambda \in \mathbb{C}, \quad y \in \mathbb{R} \tag{2.5} \]

where

\[ H_v = -\frac{d^2}{dy^2} + \tilde{W}_u(y) \tag{2.6} \]

and

\[ \tilde{W}_u(y) = -\frac{\partial f}{\partial y} + \frac{d}{dy} \frac{\partial f}{\partial y} \tag{2.7} \]
The interest of one of us (F.Z.) in the study of the asymptotic behavior of the spectrum of the Fokker-Planck operators arose in the study of a method for global optimization based on the use of suitable stochastic differential equations [11].

In §2 the eigenvalue problem for \( L_\varepsilon \) is reduced to an eigenvalue problem for a suitable Schrodinger Hamiltonian \( H_\varepsilon \). The particular Schrodinger Hamiltonian obtained when \( f \) is a fourth degree polynomial with two minimizers are studied in detail.

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

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

In §5 a theorem concerning the behavior as \( \varepsilon \to 0 \) of the difference between the resolvent of \( H_\varepsilon \) and the resolvent of the approximating Hamiltonian is proved.

Moreover the asymptotic behavior as \( \varepsilon \to 0 \) of the spectrum of \( H_\varepsilon \) and as a consequence of the spectrum of \( L_\varepsilon \) is considered.

In §6 using the Rayleigh-Ritz principle for \( H_\varepsilon \) a particularly simple asymptotic formula for the first nonzero eigenvalue of \( L_\varepsilon \) 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:

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

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

\begin{equation}
L_\varepsilon(p) = \frac{\varepsilon^2}{2} \frac{\partial^2 p}{\partial x^2} + \frac{\gamma}{\partial x} \left( \frac{f}{\partial x} p \right) \quad x \in \mathbb{R}
\end{equation}

subject to the condition

\begin{equation}
\lim_{t \to 0} p_\varepsilon(x,x_0,t) = \delta(x-x_0)
\end{equation}

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

The problem of deriving asymptotic formulas as $\varepsilon \to 0$ 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 $\varepsilon \to 0$ has received much less attention. In this paper we restrict our attention to the one-dimensional case when $L_\varepsilon$ is given by (1.4) and $f$ is a fourth degree polynomial with two minimizers.

Even in this particular case the resulting problems is an interesting singular perturbation problem for the ordinary differential operator $L_\varepsilon$. 
§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]:

\( \text{(1.1)} \quad dx(t) = -\nabla f(x(t)) dt + \epsilon \, dw(t) \)

where \( f: \mathbb{R} \to \mathbb{R} \) is a smooth function called potential, \( \mathbb{R} \) is the real line, \( \epsilon \) 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 \( x_c(t) \) is a stochastic process.

The transition probability density \( p_c(x,x_0,t) \) of \( x_c(t) \) is defined as:

\( \text{(1.2)} \quad p_c(x,x_0,t) dx = P_{x_0} \{ x_c(t) \in (x,x+dx) \mid x_c(0) = x_0 \} \)
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
References


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

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

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

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

where $\xi_{1,2} = \frac{-3b \pm \sqrt{9b^2 - 24ac}}{12a}$ 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

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

(iii) by explicit computation from (ii) it is easy to obtain the following table:
Table 2

where $c_2 = \frac{1}{2} \left( \frac{d^2 f}{dx^2} (x_1) \right) < 0$, $c_1 = \frac{1}{2} \left( \frac{d^2 f}{dx^2} (x_2) \right) > 0$. Moreover $c = 2ax_1 x_2$.

(c) from Table 2, we can deduce that the equation

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

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

(iv) $U (y)$ is bounded below by a constant independent of $y$.

(vi) $U (y)$ is given by Fig. 1.
From Proposition 2.1 and Fig. 3 it follows that as $\varepsilon \to 0$

$V_\varepsilon(y)$ approaches three independent harmonic oscillator potentials,
one with vertex at $y = 0$ and equation $4\alpha'^2 y^2 + 2\alpha^2$ and two with
vertices at $y = \pm \sqrt{\frac{\varepsilon}{\varepsilon}} \alpha$ and equations $16\alpha''(y \pm \sqrt{\frac{\varepsilon}{\varepsilon}} \alpha)^2 - 4\alpha^2$.

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

\[(2.24) \quad \lambda_n^{(1)} = 4\alpha^2(n+1) \quad n = 0, 1, 2, \ldots \]

the eigenvalues corresponding to the remaining two harmonic oscillators
are

\[(2.25) \quad \lambda_n^{(2)} = 8\alpha^2n \quad n = 0, 1, 2, \ldots \]

\[(2.26) \quad \lambda_n^{(3)} = 8\alpha^2n \quad n = 0, 1, 2, \ldots \]

In section 5 we will prove that the eigenvalues of

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

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

\[(2.28) \quad \lim_{\varepsilon \to 0} \lambda_{2n+4}(\varepsilon) = 4\alpha^2(2n+1) \quad n = 0, 1, 2 \]
as can be seen from (2.24) and
\[
\lim_{\varepsilon \to 0} - \lambda_{3+4n}(\varepsilon) = \lim_{\varepsilon \to 0} - \lambda_{4+4n}(\varepsilon) = \lim_{\varepsilon \to 0} - \lambda_{5+4n}(\varepsilon) = 8\alpha^2(n+1)
\]

\[n = 0,1,2,\ldots\]

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

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

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

\[
\lambda_n^{(1)} = (2n+1)c - c \quad n = 0,1,2,\ldots \quad (c > 0)
\]

the eigenvalues corresponding to the remaining two harmonic oscillators are

\[
\lambda_n^{(2)} = (2n+1)|c_1| - c_1 \quad n = 0,1,2,\ldots \quad (c_1 < 0)
\]

\[
\lambda_n^{(3)} = (2n+1)c_2 - c_2 \quad n = 0,1,2,\ldots \quad (c_2 > 0)
\]

In section 5 we will prove that the eigenvalues of

\[
N_\varepsilon = -\frac{d^2}{dy^2} + U_\varepsilon(y) \quad y \in \mathbb{R}
\]
approaches (2.30), (2.31), (2.32) when $\epsilon \to 0$. In particular we will show that the first eigenvalue $\lambda_1 = 0$ as $\epsilon \to 0$ has asymptotically multiplicity 2 (i.e. $\lambda'(\epsilon) \to 0$ when $\epsilon \to 0$). The remaining eigenvalues, since $e, e_1, e_2$ can be expressed in terms of $a, x_1, x_2$ as shown in Proposition 2.1 (iii), have multiplicity one if $\frac{x_1}{x_2}$ is irrational, have multiplicity one or three if $\frac{x_1}{x_2}$ is rational.
§3. The approximating hamiltonians.

Let $C_0^\infty(\mathbb{R})$ be the space of the infinitely differentiable functions of compact support. Let $h_0: \mathcal{D}(h_0) \subset L^2(\mathbb{R}) \to L^2(\mathbb{R})$ denote the self-adjoint extension of $-\frac{d^2}{dy^2}$ 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) $M_\varepsilon$ is essentially self-adjoint on $C_0^\infty(\mathbb{R})$ and is self-adjoint on $\mathcal{D}(h_0) \cap \mathcal{D}(y^b)$

(ii) $M_\varepsilon$ has compact resolvent

(iii) the eigenvalues of $M_\varepsilon$ 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) $N_\varepsilon$ is essentially self-adjoint on $C_0^\infty(\mathbb{R})$ and is self-adjoint on $\mathcal{D}(h_0) \cap \mathcal{D}(y^b)$

(ii) $N_\varepsilon$ has compact resolvent

(iii) the eigenvalues of $N_\varepsilon$ are non degenerate.

**Proof:** See [6] and [12].
Let $A_+ = \{y \mid y > \frac{\alpha \sqrt{6}}{3\epsilon}\}$, $A_0 = \{y \mid |y| < \frac{\alpha \sqrt{6}}{3\epsilon}\}$, $A_- = \{y \mid y < -\frac{\alpha \sqrt{6}}{3\epsilon}\}$

and define $V_{2\epsilon}$ as follows:

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

(3.1)

(see Fig. 5)

where

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

(3.2)

$$\beta = \frac{\pi}{2} \frac{3\epsilon}{\alpha \sqrt{6}}$$

(3.3)

$$V_0 = \frac{32}{3} \frac{\pi^6}{\alpha^6 \epsilon^2}$$

(3.4)

The function $V_{2\epsilon}$ as $\epsilon \to 0$ is an approximation to $V_\epsilon$ 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 $16\alpha^4 (y + \frac{\alpha \sqrt{2}}{\epsilon})^2 - 4\alpha^2$.

Let $0 \cdot \gamma_{1}(\epsilon) < \frac{\alpha \sqrt{6}}{3\epsilon}$, $0 \cdot \gamma_{2}(\epsilon) < \frac{1}{\epsilon^2}$ with
(3.5) \[ \lim_{\varepsilon \to 0} \bar{n}_1(\varepsilon) = \lim_{\varepsilon \to 0} \bar{n}_2(\varepsilon) = \infty \]

(3.6) \[ \lim_{\varepsilon \to 0} \varepsilon \bar{n}_1(\varepsilon) = \lim_{\varepsilon \to 0} \varepsilon \bar{n}_2(\varepsilon) = 0 \]

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

(3.7) \[ V_{2\varepsilon}(\bar{n}_1(\varepsilon)) = V_{2\varepsilon} \left( \frac{\sqrt{2}}{\varepsilon} \bar{n}_2(\varepsilon) - \bar{n}_2(\varepsilon) \right) \]

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

Let

(3.8) \[ I_1^{(\varepsilon)} = \{ y \in \mathbb{R} | \bar{n}_1(\varepsilon) < y < \frac{\sqrt{2}}{\varepsilon} \bar{n}_2(\varepsilon) \} \]

(3.9) \[ I_2^{(\varepsilon)} = \{ y \in \mathbb{R} | -\frac{\sqrt{2}}{\varepsilon} \bar{n}_2(\varepsilon) < y < -\bar{n}_1(\varepsilon) \} \]

we define

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

(see Fig. 6)

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

Let us now consider the operators

(3.11) \[ M^{(2)}_{\varepsilon} = -\frac{d^2}{dy^2} + V_{1\varepsilon} \quad y \in \mathbb{R} \]
we will use them to approximate \( M_\varepsilon \).

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

\[
M_\varepsilon^{(2)} v = \lambda v \quad y \in \mathbb{R}, \quad v \in L^2(\mathbb{R})
\]

can be reduced to the following eigenvalue problems:

\[
M_\varepsilon^{(2)} v = \lambda v \quad y \in A_+, \quad v \in L^2(A_+),
\]

\[
M_\varepsilon^{(2)} v = \lambda v \quad y \in A_0, \quad v \in L^2(A_0),
\]

\[
M_\varepsilon^{(2)} v = \lambda v \quad y \in A_-, \quad v \in L^2(A_-)
\]

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].

\[
\gamma_n^\pm = 4\alpha^2 [2n + \gamma - \frac{\alpha^2}{\gamma}] \quad n = 0, 1, 2, ...
\]

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

where \( N_{n\varepsilon} \) is a normalization constant and \( L_n^\gamma \) are the generalized Laguerre polynomials and \( \phi_n^{\pm} \) is defined for \( y > \frac{\alpha\sqrt{\varepsilon}}{3\varepsilon} \), \( \phi_n^{-} \) is defined \( y < -\frac{\alpha\sqrt{\varepsilon}}{3\varepsilon} \) and

\[
\gamma = \frac{1}{2\nu} \sqrt{4\alpha^4 + \nu^2}
\]

The eigenvalues and eigenfunctions of (3.15) are given by [15]:
\[\lambda_n = \delta^2 [n^2 + \delta(2n+1)] + 2\lambda \quad n = 0, 1, \ldots\]

\[\int_{n_{c/2}}^{n_{c/2}+1} \begin{cases} 
\cos^2 \gamma F(\gamma + \frac{n}{2}, \frac{1}{2} \sin^2 \gamma) & \text{when } n \text{ is even} \\
\cos^2 \gamma \sin^2 \gamma F(\gamma - \frac{n+1}{2}, -\frac{n+1}{2}, \frac{1}{2} \sin^2 \gamma) & \text{when } n \text{ is odd}
\end{cases}
\]

where \(F(x_1, x_2, x_3, x)\) is the hypergeometric function and \(\lambda\) is defined by the equation

\[V_n = \delta^2 \delta \lambda (-1), \quad n = 1.
\]

The eigenvalues of (3.13) are given by \(\lambda_{n_{c/2}}\) and \(\lambda_{n_{c/2}+1}\) for \(n = 0, 1, 2, \ldots\). The eigenvalues \(\lambda_{n_{c/2}}\) have multiplicity two. Moreover, as \(\delta \to 0\), \(\lambda_{n_{c/2}}\) approaches the eigenvalues (2.24), (2.25), (2.26) of the three harmonic oscillators considered before.

The eigenfunctions of (3.14) satisfy

\[\lambda_{n_{c/2}}(\delta \frac{\partial}{\partial \gamma}) = \frac{\partial}{\partial \gamma} (\delta \frac{\partial}{\partial \gamma}) = 0
\]

so that corresponding eigenfunctions of (3.15) can be obtained extending \(\lambda_{n_{c/2}}(\gamma)\) with zero for \(\gamma \not\in A_k\). 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 \(\lambda_{n_{c/2}}\) of (3.13) have multiplicity two, the eigenfunction of (3.13) can be chosen to be even or odd.

Let \(C_k^c \mathcal{R} = (\pm \frac{\alpha_k}{\delta} \mathcal{R}) = (\mathcal{R} \text{ is } C^c \text{ and of compact support})\) and is zero in a neighborhood of \(\gamma = \pm \frac{\alpha_k}{\delta}\) and \(\gamma = \pm \frac{3\alpha_k}{\delta}\). We have:

\[\text{Theorem 3.7. } \mathcal{M}_{(c)}^{(2)} \text{ is essentially self-adjoint on } C_k^c \mathcal{R} = (\frac{\alpha_k}{\delta} \mathcal{R}).\]

\[\text{Proof. It is a straightforward modification of Theorem 2. Appendix 2.}\]
Theorem 3.4. $M^{(1)}_\varepsilon$ is essentially self-adjoint on $C^\infty_0(\mathbb{R})$.

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

Let $\mathcal{X}_+ = \{ y \mid y > 1 \}$, $\mathcal{X}_0 = \{ y \mid 2y_1 - 1 < y < 1 \}$, $\mathcal{X}_- = \{ y \mid y < 2y_1 - 1 \}$ and define $U_\varepsilon$ as follows:

$$
U_\varepsilon = \begin{cases} 
\frac{c_0^2}{4} \left( y - 1 \right)^2 - c_1 & \text{when } y \in \mathcal{X}_+ \\
\frac{\bar{\n}_0}{\cos^2 \bar{c}(y - y_1)} - \bar{\n}_0 - c_1 & \text{when } y \in \mathcal{X}_0 \\
\frac{c_1}{4} \left( y - (2y_1 - 1) \right)^2 - c & \text{when } y \in \mathcal{X}_-
\end{cases}
$$

(see Fig. 7)

where

$$
(3.24) \quad \frac{1}{2y_2} = (y_2 - 1) \\
(3.25) \quad \frac{1}{2y_1} = (2y_1 - 1) \\
(3.26) \quad \bar{c} = \frac{1}{2} \frac{1}{\varepsilon} \\
(3.27) \quad \bar{\n}_\varepsilon = \frac{c_1}{\bar{c}}
$$

Let us remember that $y_1, y_2, \varepsilon_1, \varepsilon_2$ depend on $\varepsilon$ (Proposition 2.2).
Since equation (3.7) implies that \( \lim_{c \to 0} \frac{\tilde{\eta}_1(c)}{\tilde{\eta}_2(c)} = \text{constant} \neq 0 \) from (4.55), (4.56) and (4.39) we have

\[
(4.57) \quad |\hat{V}_{1c}^{\nu}(y)| \leq \frac{1}{4\nu^2} \frac{96\alpha^*}{|y - \frac{\alpha\sqrt{\beta}}{\tilde{3}\tilde{c}}|^5} \leq 96\alpha^* \frac{\sqrt{2\nu}}{|1 - \tilde{\eta}_2/\sqrt{2\nu}|^5}
\]

Reasoning in the same way it can be shown that:

\[
(4.58) \quad |F_c(y)| \leq \text{constant } \epsilon^{1-3\delta_1}
\]

when \( |y - \frac{\alpha\sqrt{\beta}}{\epsilon} | < \tilde{\eta}_2(\epsilon) \).

Let us prove (4.44). From Proposition 2.1 (i) we know that \( \hat{V}_c \) 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 \( y = \pm \frac{1}{\epsilon} \left( \frac{4\epsilon^2 - \sqrt{4\epsilon^4 + 9\epsilon^2}}{3} \right)^{1/2} \). Moreover

\[
(4.60) \quad \lim_{\epsilon \to 0} \epsilon^3 \hat{V}_c \left( \pm \frac{1}{\epsilon} \left( \frac{4\epsilon^2 + \sqrt{4\epsilon^4 + 9\epsilon^2}}{3} \right)^{1/2} \right) = \frac{32}{27} \alpha^6
\]

\[
(4.61) \quad \lim_{\epsilon \to 0} \hat{V}_c \left( \pm \frac{1}{\epsilon} \left( \frac{4\epsilon^2 - \sqrt{4\epsilon^4 + 9\epsilon^2}}{3} \right)^{1/2} \right) = -4\alpha^2 + \tilde{c}
\]

and for \( 0 < \epsilon < \epsilon_0 \)

\[
(4.62) \quad \frac{1}{\epsilon} \left( \frac{4\epsilon^2 - \sqrt{4\epsilon^4 + 9\epsilon^2}}{3} \right)^{1/2} \in 1^{(\epsilon)}_1(c), \quad -\frac{1}{\epsilon} \left( \frac{4\epsilon^2 + \sqrt{4\epsilon^4 + 9\epsilon^2}}{3} \right)^{1/2} \in 1^{(\epsilon)}_1(c)
\]
When \( |y| < \bar{r}_1(\varepsilon) = \varepsilon^{-\delta_1} \) we have

\[
\hat{V}_c'''(y) = |120 \varepsilon^4 y^3 - 96 \varepsilon^2 y| \leq 24 \varepsilon^{2-\delta_1} (5 \varepsilon^{2-\delta_1} \alpha' + 4 \alpha')
\]

and

\[
\hat{V}_c'''(y) = 32 \varepsilon^2 |\cos \bar{y}| |\sin \bar{y}| (2 + \sin \bar{y})
\]

\[
\leq \frac{\varepsilon^{2-\delta_1}}{\cos^2 \left(\frac{\pi}{2} - \frac{3\bar{y}}{\varepsilon^{1-\delta_1}}\right)}
\]

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

\[
|F_c(y)| \leq \text{constant } \varepsilon^{1-\delta_1}
\]

When \( |y - \frac{\alpha \sqrt{2}}{\varepsilon}| < \bar{r}_2(\varepsilon) \) we have \( \hat{V}_1c(y) = \hat{V}_2c(y) \) so using the Taylor formula at \( y = \frac{\alpha \sqrt{2}}{c} \) we have

\[
F_c(y) = -6 \alpha \sqrt{2} \varepsilon \left(y - \frac{\alpha \sqrt{2}}{c}\right) - 3 \varepsilon^2 \left(y - \frac{\alpha \sqrt{2}}{c}\right)^2 + \frac{P''''(z)}{3} \left(y - \frac{\alpha \sqrt{2}}{c}\right)^3
\]

with \( z \) an intermediate point in the interval \( \left[\frac{\alpha \sqrt{2}}{c}, y\right] \)

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

\[
|\hat{V}_c'''(y)| \leq 144 \varepsilon^{2-\delta_1} \alpha' + 624 \varepsilon^2 \bar{r}_2^2 + 360 \varepsilon \alpha \varepsilon^{3\bar{r}_2} + 120 \varepsilon^{2\bar{r}_2}
\]

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 \( \hat{V}_{1_{\varepsilon}} \equiv \hat{V}_{2_{\varepsilon}} \). The proof of (4.41) follows from the fact that

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

and

\[
0 \leq \hat{V}_{2_{\varepsilon}}^{-1}(\hat{V}_{2_{\varepsilon}} - \hat{V}_{1_{\varepsilon}}) = 1 - \frac{\hat{V}_{1_{\varepsilon}}}{\hat{V}_{2_{\varepsilon}}} \leq 1
\]

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

\[
\hat{V}_{1_{\varepsilon}}(y) = \hat{V}_{2_{\varepsilon}}(\bar{\eta}_1(\varepsilon)) = V_0 \tan^2 \beta \bar{\eta}_1(\varepsilon) + 2a^2 + \tilde{c}
\]

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

\[
\hat{V}_{1_{\varepsilon}}(y) \geq \text{constant} \quad \varepsilon^{-2} \bar{\eta}_1
\]

for \( 0 < \varepsilon < \varepsilon_0 \).

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

\[
F_{\varepsilon}(y) = \hat{V}_{\varepsilon}(y) - \hat{V}_{1_{\varepsilon}}(y)
\]

Using the Taylor's formula at \( y = 0 \) we have

\[
F_{\varepsilon}(y) = -3\varepsilon^2 y^2 + \frac{\hat{V}'''(\xi)}{3!} y^3
\]

with \( \xi \) 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

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

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

$$\tilde{\eta}_1(\varepsilon) = \varepsilon^{-\delta_1} \quad 0 < \delta_1 < \frac{1}{3}$$

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

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

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

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

$$\left\| \hat{V}_{1\varepsilon}^{-1} P_1 \right\| \leq \text{constant} \varepsilon^{2\delta_1}$$

$$\left\| \left( \hat{V}_{2\varepsilon} - \hat{V}_{1\varepsilon} \right)(I - P_2) \right\| \leq \text{constant} \varepsilon^{1 - 3\delta_1}$$

$$\left\| \hat{V}_{1\varepsilon}^{-1} P_2 \right\| \leq \text{constant}$$

$$\left\| \hat{V}_{2\varepsilon}^{-1} P_1 \right\| \leq \text{constant} \varepsilon^{2\delta_1}$$

where $I$ is the identity on $L^2(\mathbb{R})$ and $\| \cdot \|$ is the operator norm induced by the $L^2$ norm.
(4.33) \((\hat{M} + z)^2 \geq \hat{V}^2\) on \(C^\omega_0(\mathbb{R}) \times C^\omega_0(\mathbb{R})\)

(4.34) \((\hat{M}^{(1)} + z)^2 \geq \hat{V}^2\) on \(C^\omega_0(\mathbb{R}) \times C^\omega_0(\mathbb{R})\)

(4.35) \((\hat{M}^{(2)} + z)^2 \geq \hat{V}^2\) on \(C^\omega_0(\mathbb{R}) \times C^\omega_0(\mathbb{R})\)

where \(0 < \tilde{\beta} < 1\)

**Proof:** It follows from Theorem 4.1 since \(\hat{V}_1 = \hat{V}_2 = \hat{V}\) and \(\hat{c} > 0\) and the similar statements for \(\hat{V}_1, \hat{V}_2, \hat{V}_3, \hat{V}_4\).

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

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

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

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

**Proof:** Note that (4.27), (4.28), (4.29) imply \(\hat{V}_1, \hat{V}_2, \hat{V}_2 \geq \text{constant} > 0\) so that \(\hat{V}_1, \hat{V}_2, \hat{V}_2\) are bounded operators. The proof of Theorem 4.3 follows immediately from Theorem 2.21, page 330 of Kato [17].

**Definition 4.4.** Let \(P_1\) be the projection on the subspace of the functions of \(L^2(\mathbb{R})\) that have support on \(I_1^{(\varepsilon)} \cup I_2^{(\varepsilon)}\). That is \(P_1\) is the multiplication operator given by \(\chi_{I_1^{(\varepsilon)} \cup I_2^{(\varepsilon)}}\).
(4.26) \[(1 - \chi_1^\varepsilon_1)(1 - \chi_2^\varepsilon_2) \{((1 - \overline{\delta})V_2^\varepsilon + 2zV_2^\varepsilon + z^2 - \overline{V}_2^\varepsilon) + \]
\[+ \chi_1^\varepsilon_1 \{((1 - \overline{\delta})V_2^\varepsilon_2(\overline{\eta}_1) + 2zV_2^\varepsilon_2(\overline{\eta}_1) + z^2) + \]
\[+ \overline{c}_1\{\delta(y - \overline{\eta}_1) + \delta(y + \overline{\eta}_1)\} + \overline{c}_2\{\delta(y - \frac{\alpha \sqrt{2}}{\varepsilon} + \overline{\eta}_2) + \delta(y + \frac{\alpha \sqrt{2}}{\varepsilon} - \overline{\eta}_2)\} \geq 0.\]

In fact for \(z > z_0 > 0, 0 < \varepsilon < \varepsilon_0\) we have \(((1 - \overline{\delta})V_2^\varepsilon + 2zV_2^\varepsilon + z^2 - \overline{V}_2^\varepsilon) \geq 0.\) Moreover
\[((1 - \overline{\delta})V_2^\varepsilon(\overline{\eta}_1) + 2zV_2^\varepsilon(\overline{\eta}_2) + z^2) \geq 0\) and \(\overline{c}_1 \geq 0, \overline{c}_2 \geq 0.\)

The estimate (4.2) is established.

Let \(\hat{c}\) be a constant such that

\[(4.27) \quad \hat{V}_1^\varepsilon = V_1^\varepsilon + \hat{c} > 0 \quad \text{and} \quad (V_1^\varepsilon + \hat{c})^2 \geq V_1^\varepsilon\]
\[(4.28) \quad \hat{V}_1^\varepsilon = V_1^\varepsilon + \hat{c} > 0 \quad \text{and} \quad (V_1^\varepsilon + \hat{c})^2 \geq V_1^\varepsilon\]
\[(4.29) \quad \hat{V}_2^\varepsilon = V_2^\varepsilon + \hat{c} > 0 \quad \text{and} \quad (V_2^\varepsilon + \hat{c})^2 \geq V_2^\varepsilon\]

We define

\[(4.30) \quad \hat{M}^\varepsilon = -\frac{d^2}{dy^2} + \hat{V}_1^\varepsilon = M^\varepsilon + \hat{c}\]
\[(4.31) \quad \hat{M}^{(1)}_1 = -\frac{d^2}{dy^2} + \hat{V}_1^\varepsilon = M^{(1)}_1 + \hat{c}\]
\[(4.32) \quad \hat{M}^{(2)}_1 = -\frac{d^2}{dy^2} + \hat{V}_2^\varepsilon = M^{(2)}_1 + \hat{c}\]

**Theorem 4.2.** There exist \(z_0 > 0\) and \(\varepsilon_0 > 0\) such that for \(z \geq z_0\)

and \(0 < \varepsilon < \varepsilon_0\) we have:
The estimate (4.3) has been established.

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

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

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

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

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

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

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

$$(4.24) \quad \tilde{c}_1 = 2\beta V_0 \cos^3 \beta \tilde{n}_1 \sin \beta \tilde{n}_1 \geq 0$$

$$(4.25) \quad \tilde{c}_2 = \frac{8\alpha \varepsilon}{\sqrt{2}\lambda} \left| 1 - \sqrt{2} \sqrt{\tilde{n}_1} \frac{1}{(1 - \sqrt{2} \sqrt{\tilde{n}_1})^3} \right| \geq 0$$

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

Since $V_{1\varepsilon} = V_{2\varepsilon}$ when $y \notin \mathbb{R}\setminus I_1(\varepsilon) \cup I_2(\varepsilon)$, we can rewrite equation (4.22) as follows:
since \( V_0 \) given (3.4) goes to infinity when \( \varepsilon \to 0 \). The last inequality in (4.16) holds \( \forall z > 0 \), \( 0 < \varepsilon < \frac{2}{3} \frac{\alpha^2}{\pi} (6(1+\bar{U}))^{1/4} \).

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

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

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

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

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

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

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

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

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

The proof of (4.3) for \( y \in A_+ \) is analogous to the proof given for \( y \in A_- \) and will be omitted.
and this last expression can be made positive for \( z > z_0 \) and \( 0 < \varepsilon < \varepsilon_0 \). Choosing \( z_0 \) and \( \varepsilon_1 \). The estimate (4.1) is established.

Let us now prove (4.3). Let \( 0 < \varepsilon < 1 \) proceeding as we have done proving (4.1), we obtain as a form on \( C^\infty_0(\mathbb{R} - \{ z + \frac{\alpha \sqrt{6}}{\varepsilon} \}) \times C^\infty_0(\mathbb{R} - \{ z - \frac{\alpha \sqrt{6}}{\varepsilon} \}) \)

\[
(M^{(2)}_c + z)^2 - \varepsilon V^2_{z \varepsilon} \geq (1 - \varepsilon) V^2_{z \varepsilon} + 2zV_{z \varepsilon} + z^2 - V''_{z \varepsilon}
\]

(4.12)

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

\[
(1 - \varepsilon) V^2_{z \varepsilon} + 2zV_{z \varepsilon} + z^2 - V''_{z \varepsilon} \geq 0 \quad y \in \mathbb{R}.
\]

(4.13)

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

\[
z^2 + 2(V_0 \tan \gamma + \alpha^2)z + [(1 - \varepsilon)(V_0 \tan \gamma + \alpha^2)^2 + -8a^2 \frac{2 \sin^2 \gamma + 1}{\cos^2 \gamma}]
\]

(4.14)

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

\[
z^2 + 4a^2z - (60 + \Delta \varepsilon) \geq 0
\]

(4.15)

when \( z > (\sqrt{64 + \Delta \varepsilon} - 2)\alpha^2 \) and \( \forall \varepsilon > 0 \).

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

\[
z^2 + 4a^2z + \frac{1}{\cos^2 \gamma} [(1 - \varepsilon)V_0 \sin^2 \gamma - 8a^2(2 \sin^2 \gamma + 1)] \geq 0
\]

(4.16)

\[
\geq z^2 + 4a^2z + 4 \frac{1 - \varepsilon}{4} V_0^2 - 24a^2 \geq 0
\]
(4.7) \[ F_1(y^2) = 2zV_e + z^2 - \nabla_w^2 y \geq 0 \quad y \in \mathbb{R} \]

Let us define

\[ t = y^2 \]

\[ A = 2z\varepsilon^4 \]

\[ B = 8z\varepsilon^2 + 30\varepsilon^4 \]

\[ C = 2z(4\varepsilon^4 - 3\varepsilon^2) + 48\varepsilon^2 \]

\[ D = 4z\varepsilon^2 + z^2 - 8\varepsilon^2 + 6\varepsilon^2 \]

A simple computation shows that

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

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

(4.9) \[ At^2 - Bt + C \]

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

where

(4.10) \[ At_0^2 - Bt_0 + C = \frac{4AC - B^2}{4A} = -\frac{\varepsilon^2}{2z} (225\varepsilon^2 + 244\varepsilon^2 + 122\varepsilon^2) \leq 0 \]

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

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

So that \( \forall t \geq 0 \)

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

\[ = z^2 - 20\alpha^2z - 56\alpha^w - 84\varepsilon^2 - 630 \frac{\alpha^2 \varepsilon^2}{z} - \frac{3375}{2} \frac{\varepsilon^6}{z^2} \]
§4. The basic estimates.

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

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

\((4.1)\quad (M_\varepsilon + z)^2 \geq V_\varepsilon^2\) on $C^\infty_0(\mathbb{R}) \times C^\infty_0(\mathbb{R})$

\((4.2)\quad (M_\varepsilon^{(1)} + z)^2 \geq |V_\varepsilon|\) on $C^\infty_0(\mathbb{R}) \times C^\infty_0(\mathbb{R})$

\((4.3)\quad (M_\varepsilon^{(2)} + z)^2 \geq \tilde{V}_\varepsilon^2\) on $C^\infty_0(\mathbb{R} - \{ \pm \frac{\sqrt{6\varepsilon}}{3\varepsilon} \}) \times C^\infty_0(\mathbb{R} - \{ \pm \frac{\sqrt{6\varepsilon}}{3\varepsilon} \})$

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

Proof: Let us first prove (4.1) and let $p = i \frac{d}{dy}$. Then as a form on $C^\infty_0(\mathbb{R}) \times C^\infty_0(\mathbb{R})$ we have:

\[(4.4)\quad (M_\varepsilon + z)^2 = (p^2 + V_\varepsilon + z)^2 = p^2 + V_\varepsilon^2 + 2zV_\varepsilon + z^2 + 2p(V_\varepsilon + z)p - V_\varepsilon''\]

Since $V_\varepsilon$ is constant independent of $\varepsilon$ when $0 < \varepsilon < \varepsilon_0$, so

\[(4.5)\quad p(V_\varepsilon + z)p \geq 0\) on $C^\infty_0(\mathbb{R}) \times C^\infty_0(\mathbb{R})$

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''\) on $C^\infty_0(\mathbb{R}) \times C^\infty_0(\mathbb{R})$

To prove (4.1) it will be enough to show that for $z > z_0$ and $0 < \varepsilon < \varepsilon_0$ we have:
Proceeding as before let us now consider

\( \begin{align*} 
N_{c}^{(2)} &= -\frac{d^2}{dy^2} + U_{2c} \quad y \in \mathbb{R} \\
N_{c}^{(1)} &= -\frac{d^2}{dy^2} + U_{1c} \quad y \in \mathbb{R} 
\end{align*} \)

we will use them to approximate \( N_{c} \).

The eigenvalue problem for \( N_{c}^{(2)} \) can be solved analogously to

the eigenvalue problem for \( M_{c}^{(2)} \) in particular as \( \varepsilon \to 0 \) the eigenvalues of \( N_{c}^{(2)} \) approach the eigenvalues (2.30), (2.31), (2.32) of the

three harmonic oscillators considered before.

Let \( C_{c}^{\infty}(\mathbb{R} - \{ 2y_{1}, -n_{2} \} - \{ n_{2} \}) = \{ f \mid f \text{ is } C^{\infty} \text{ and of compact support and is zero in a neighborhood of } y = 2y_{1} - n_{2} \text{ and } y = n_{2} \} \).

We have:

**Theorem 3.5.** \( N_{c}^{(2)} \) is essentially self-adjoint on \( C_{c}^{\infty}(\mathbb{R} - \{ 2y_{1}, -n_{2} \} - \{ n_{2} \}) \).

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

**Theorem 3.6.** \( N_{c}^{(1)} \) is essentially self-adjoint on \( C_{c}^{\infty}(\mathbb{R}) \).

**Proof:** It follows immediately from Theorem 10.23 page 315 of Weidmann [10].
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 $\varepsilon \to 0$ is an approximation to $U_\varepsilon$. 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

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

two intervals such that

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

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 $J_1(\varepsilon) \cap J_2(\varepsilon) = \{\phi\}$, note that because of symmetry $U_{2\varepsilon}(y_1 - \mu_2(\varepsilon)) = U_{2\varepsilon}(y_1 + \mu_2(\varepsilon))$

finally later we will need

\[ (3.30) \quad \lim_{\varepsilon \to 0} U_{2\varepsilon}(y_1 + \mu_2(\varepsilon)) = \infty. \]

Let us define

\[ (3.31) \quad U_{1\varepsilon}(y) = \begin{cases} U_{2\varepsilon}(y) & y \not\in J_1(\varepsilon) \cup J_2(\varepsilon) \\ U_{2\varepsilon}(y_1 + \mu_2(\varepsilon)) & y \in J_1(\varepsilon) \\ U_{2\varepsilon}(y_1 - \mu_2(\varepsilon)) & y \in J_2(\varepsilon) \end{cases} \]

(see Fig. 8)
Let \( y \in I_1^{(e)} \cup I_2^{(e)} \) then \( \hat{V}_{1e}(y) = V_{2e}(\bar{\eta}_1(e)) + \hat{c} \) so that

\[
|\hat{V}_{1e}^{-1}(\hat{V}_c - \hat{V}_{1e})| = |1 - \frac{\hat{V}_{1e}}{\hat{V}_c}| \leq 1 + \frac{V_{2e}(\bar{\eta}_1(e)) + \hat{c}}{m(e)}
\]

where

\[
m(e) = \min_{y \in I_1^{(e)} \cup I_2^{(e)}} \hat{V}_e(y) = \min_{y \in I_1^{(e)}} \hat{V}_e(y) = \min(\hat{V}_e(\bar{\eta}_1(e)), \hat{V}_e(\frac{\alpha_0}{e} - \bar{\eta}_2(e)));
\]

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

An elementary computation now shows that

\[
|\hat{V}_{1e}^{-1}(\hat{V}_c - \hat{V}_{1e})| \leq \text{constant, for } 0 < e < e_* \text{ when } y \in I^{(e)} \cup I^{(e)}.
\]

Let \( y > \frac{2\alpha_0}{e} + \bar{\eta}_2(e) \) we have \( V_{1e}(y) = V_{2e}(y) \). Define \( y' = y - \frac{2\alpha_0}{5e} \)

so that we have \( y' > \bar{\eta}_2(e) + \frac{1}{\sqrt{2e}} \) and

\[
V_{1e}(y) = 4\alpha^*(y' - \frac{1}{\sqrt{2e}y'})^2 - 4\alpha^2
\]

\[
= 4\alpha^*(y' - \frac{1}{\sqrt{2e}})^2(1 + \frac{1}{y', \sqrt{2e}})^2 - 4\alpha^2
\]

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

\[
V_{1e}(y) \leq 10\alpha^*(y - \frac{\alpha \sqrt{e}}{c})^2 - 4\alpha^2 \text{ when } y > \frac{\alpha \sqrt{e}}{c} + \bar{\eta}_1(e).
\]
Moreover

\[(4.09) \quad V_c(y) = -4\alpha^2 - 6\sqrt{2} \omega \epsilon(y - \frac{\alpha \sqrt{2}}{\epsilon}) + (16\alpha^4 - 3\epsilon^2)(y - \frac{\alpha \sqrt{2}}{\epsilon})^2 + \]

\[+ 24\sqrt{2} \omega \epsilon^3 (y - \frac{\alpha \sqrt{2}}{\epsilon})^3 + 26 \epsilon^2\epsilon^2 (y - \frac{\alpha \sqrt{2}}{\epsilon})^4 + \]

\[+ 6\sqrt{2} \omega \epsilon^5 (y - \frac{\alpha \sqrt{2}}{\epsilon})^5 + \epsilon^6 (y - \frac{\alpha \sqrt{2}}{\epsilon})^6 \]

From (4.08), (4.09) when \(0 < \epsilon < \epsilon_0\) and \(y \geq \frac{\alpha \sqrt{2}}{\epsilon} + \eta_2(\epsilon)\) we have:

\[(4.70) \quad \left| \frac{V_{1r}}{V_c} \right| \leq \frac{10\alpha^4 + 4}{\eta_2(\epsilon)} \left| - \frac{4 \epsilon^2}{\eta_2(\epsilon)} - \frac{6\sqrt{2}\omega \epsilon}{\eta_2(\epsilon)} + 16\alpha^2 - 3\epsilon^2 \right| \]

so that \(\left| \frac{V_{1r}}{V_c} \right| \leq \text{constant} \) when \(0 < \epsilon < \epsilon_0\). That is

\[(4.71) \quad \left| \hat{V}_{1r}^{-1}(\hat{V}_c \hat{V}_{1r}) \right| \leq \text{constant} \quad \text{when} \quad 0 < \epsilon < \epsilon_0, \quad y > \frac{\alpha \sqrt{2}}{\epsilon} + \eta_2(\epsilon). \]

Reasoning in the same way it can be shown that

\[(4.72) \quad \left| \hat{V}_{1r}^{-1}(\hat{V}_c \hat{V}_{1r}) \right| < \text{constant} \quad \text{when} \quad 0 < \epsilon < \epsilon_0, \quad y < - \frac{\alpha \sqrt{2}}{\epsilon} - \eta_2(\epsilon). \]

The equations (4.06), (4.71), (4.72) establish (4.44).

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

\[(4.73) \quad \hat{V}_{1r}(y) = V_{2r}(\eta_1(\epsilon)) + \epsilon \]
From (4.39) it follows that:

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

Moreover

\[(4.76) \quad V_{1c}(y) = V_{2c}(y) \geq V_{2c}\left(\frac{\alpha\sqrt{T}}{c} + \widetilde{h}_2(\varepsilon)\right) \quad \text{when } 0 < \varepsilon < \varepsilon_0, \ y > \frac{\alpha\sqrt{T}}{c} + \widetilde{h}_2(\varepsilon)\]

and

\[(4.77) \quad \lim_{\varepsilon \to 0} \tilde{h}_1^2(\varepsilon) V_{2c}\left(\frac{\alpha\sqrt{T}}{c} + \widetilde{h}_2(\varepsilon)\right) = \text{constant} \neq 0\]

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

\[(4.78) \quad |\tilde{V}_{1c}^{-1}| < \text{constant } \varepsilon^{2\delta_1} \quad \text{when } 0 < \varepsilon < \varepsilon_0, \ y > \frac{\alpha\sqrt{T}}{c} + \widetilde{h}_2(\varepsilon)\]

Reasoning in the same way it can be shown that

\[(4.79) \quad |\tilde{V}_{1c}^{-1}| < \text{constant } \varepsilon^{2\delta_1} \quad \text{when } 0 < \varepsilon < \varepsilon_0, \ y < -\frac{\alpha\sqrt{T}}{c} - \widetilde{h}_1(\varepsilon).\]

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

This completes the proof of Theorem 4.0.
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:

\begin{align*}
(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_{z_\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_{z_\varepsilon}^2 \quad \text{on } C_0^\infty(\mathbb{R} - \{2y_1 - n_2\} - \{n_2\}) \times C_0^\infty(\mathbb{R} - \{2y_1 - n_2\})
\end{align*}

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

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

\begin{align*}
(4.83) & \quad (N_\varepsilon + z)^2 - U_\varepsilon^2 \geq 2zU_\varepsilon + z^2 - U_\varepsilon''
\end{align*}

So that to prove (4.80) it will be enough to show that for

$z > z_0$, $0 < \varepsilon < \varepsilon_0$ we have

\begin{align*}
(4.84) & \quad F_\varepsilon(y) = 2zU_\varepsilon + z^2 - U_\varepsilon'' \geq 0 \quad y \in \mathbb{R}
\end{align*}

A simple computation shows that

\begin{align*}
(4.85) & \quad F_\varepsilon(y) = 2a^2 \varepsilon^4 y^6 + \frac{6ab}{\sqrt{2}} \varepsilon^3 y^5 + (2z \frac{9}{8} b^2 + 2ac) \varepsilon^2 - 50a^2 \varepsilon^4 y^4 + \\
& \quad + \frac{6}{\sqrt{2}} bcz \varepsilon y^3 + (2z(\frac{9}{8} b^2 + 2ac) \varepsilon^2 - 12(\frac{9}{8} b^2 + 2ac) \varepsilon^2) y^2 + \\
& \quad - \frac{6bcz}{\sqrt{2}} \varepsilon + \frac{18bc}{\sqrt{2}} \varepsilon y - 2zc - 2c^2 + 6ac^2 + z^2
\end{align*}
Let $t = ey$, rearranging the terms in (4.85) we have

\[(4.86) \quad F_2(\frac{t}{c}) = 2z\left\{\left[\frac{1}{e^2} - \frac{1}{4} t^2(2at^2 + \frac{3b}{\sqrt{2}} t + 2c)^2 - (3at^2 + \frac{3b}{\sqrt{2}} t + c) + \frac{z}{2}\right] + \right.\]

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

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 $\hat{c}_e$ be a constant such that

\[(4.87) \quad \hat{U}_e = U_e + \hat{c}_e, \quad \text{and} \quad (U_e + \hat{c}_e)^2 \geq U_e^2.\]

\[(4.88) \quad \hat{U}_{1e} = U_{1e} + \hat{c}_e, \quad \text{and} \quad (U_{1e} + \hat{c}_e)^2 \geq U_{1e}^2.\]

\[(4.89) \quad \hat{U}_{2e} = U_{2e} + \hat{c}_e, \quad \text{and} \quad (U_{2e} + \hat{c}_e)^2 \geq U_{2e}^2.\]

We define

\[(4.90) \quad \hat{N}_e = - \frac{d^2}{dy_e^2} + \hat{U}_e = N_e + \hat{c}_e.\]

\[(4.91) \quad \hat{N}_{1e} = - \frac{d^2}{dy_{1e}^2} + \hat{U}_{1e} = N_{1e} + \hat{c}_e.\]

\[(4.92) \quad \hat{N}_{2e} = - \frac{d^2}{dy_{2e}^2} + \hat{U}_{2e} = N_{2e} + \hat{c}_e.\]
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

\[
(4.93) \quad (\hat{N}_\varepsilon + z)^2 \geq \hat{U}_\varepsilon^2 \quad \text{on} \quad C_0^\infty(\mathbb{R}) \times C_0^\infty(\mathbb{R})
\]
\[
(4.94) \quad (\hat{N}_\varepsilon^{(1)} + z)^2 \geq \bar{\beta} \hat{U}_{1\varepsilon}^2 \quad \text{on} \quad C(\mathbb{R}) \times C_0^\infty(\mathbb{R})
\]
\[
(4.95) \quad (\hat{N}_\varepsilon^{(z)} + z)^2 \geq \bar{\beta} \hat{U}_{2\varepsilon}^2 \quad \text{on} \quad C_0^\infty(\mathbb{R} \setminus \{2y_1 - \eta_z\}) \times C_0^\infty(\mathbb{R} \setminus \{2y_1 - \eta_z\} \setminus \{\nu_2\})
\]

where \( 0 < \bar{\beta} < 1 \).

Proof: It follows from Theorem 4.7 since \( \hat{U}_\varepsilon'' = U'' \), \( \hat{U}_\varepsilon^2 \geq U_\varepsilon^2 \) 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 \( z > z_0 \) and \( 0 < \varepsilon < \varepsilon_0 \) we have:

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

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 \( J_1^{(c)} \cup J_2^{(c)} \).

**Definition 4.11.** Let \( P_2^* \) be the projection on the subspace of the functions of \( L^2(\mathbb{R}) \) that have support on \( \mathbb{R}\setminus U_1^{(c)} \) where
\[
U_1^{(c)} = \{ y \mid |y| < \mu_1(\varepsilon) \} \cup \{ y \mid |y-y_1| < \mu_2(\varepsilon) \} \cup \{ y \mid |y-y_2| < \mu_3(\varepsilon) \}
\]

Let us now choose
\[
(4.99) \quad \mu_1(\varepsilon) = \varepsilon^{-\delta_1} \quad 0 < \delta_1 < \frac{1}{3}
\]

\( \mu_2(\varepsilon) \) 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:
\[
(4.100) \quad \| (\hat{U}_{1\varepsilon} - \hat{U}_{1c}) (I-P_1^*) \| = 0
\]
\[
(4.101) \quad \| \hat{U}_{1\varepsilon}^{-1} (\hat{U}_{1\varepsilon} - \hat{U}_{1c}) P_1^* \| \leq \text{constant}
\]
\[
(4.102) \quad \| \hat{U}_{1\varepsilon}^{-1} P_1^* \| \leq \text{constant} \varepsilon^{2\delta_1}
\]
\[
(4.103) \quad \| (\hat{U}_{1\varepsilon} - \hat{U}_{1c}) (I-P_2^*) \| \leq \text{constant} \varepsilon^{1-\delta_1}
\]
\[
(4.104) \quad \| \hat{U}_{1\varepsilon}^{-1} (\hat{U}_{1\varepsilon} - \hat{U}_{1c}) P_2^* \| \leq \text{constant}
\]
\[
(4.105) \quad \| \hat{U}_{1\varepsilon}^{-1} P_2^* \| \leq \text{constant} \varepsilon^{2\delta_1}
\]

**Proof:** The estimates (4.100), (4.101), \ldots, (4.105) can be proved has the corresponding estimates (4.40), (4.41), \ldots, (4.45) of Theorem 4.6.
§5. The behavior as \( \varepsilon \to 0 \) of eigenvalues and eigenvectors of \( \hat{M}_c, \hat{N}_c \).

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

Theorem 5.1. There exist constants \( A, z_o > 0, \varepsilon_o > 0, \varepsilon_1^* > 0 \) such that for \( z > z_o, \ 0 < \varepsilon < \varepsilon_o \) we have

\[
\| (\hat{M}_c^{(2)} + z)^{-1} - (\hat{M}_c + z)^{-1} \| \leq A \varepsilon_1^*
\]

\[
\| (\hat{M}_c^{(2)} + z)^{-1} - (\hat{M}_c^{(1)} + z)^{-1} \| \leq A \varepsilon_1^*
\]

\[
\| (\hat{M}_c^{(1)} + z)^{-1} - (\hat{N}_c + z)^{-1} \| \leq A \varepsilon_1^*
\]

\[
\| (\hat{N}_c^{(2)} + z)^{-1} - (\hat{N}_c + z)^{-1} \| \leq A \varepsilon_1^*
\]

\[
\| (\hat{N}_c^{(2)} + z)^{-1} - (\hat{N}_c^{(1)} + z)^{-1} \| \leq A \varepsilon_1^*
\]

\[
\| (\hat{N}_c^{(2)} + z)^{-1} - (\hat{N}_c^{(1)} + z)^{-1} \| \leq A \varepsilon_1^*
\]

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 \( \hat{M}_c \) converges to the resolvent of \( \hat{M}_c^{(2)} \) and the resolvent of \( \hat{N}_c \) converges to the resolvent of \( \hat{N}_c^{(2)} \) as \( \varepsilon \to 0 \). In section 3 we have
studied the eigenvalues and eigenfunctions of $M^{(2)}_\varepsilon$ and $N^{(2)}_\varepsilon$; here we will see the consequences of (5.1), (5.4) on the eigenvalues and of $M^{(2)}_\varepsilon$, $N^{(2)}_\varepsilon$.

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

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

\begin{align}
\lambda^+_{n\varepsilon} &= 8\alpha^2 n + O(\varepsilon^2) \quad n = 0, 1, 2, \ldots \\
\lambda^0_{n\varepsilon} &= 4\varepsilon^2 (n+1) + O(\varepsilon^2) \quad n = 0, 1, 2, \ldots
\end{align}

(see Fig. 9).

\begin{figure}[h]
\centering
\begin{tikzpicture}
  \draw[->] (-3,0) -- (6,0);
  \foreach \x in {0,1,2,3,4,5,6}
  \draw[fill] (\x,0) circle (2pt) node[below] {$\lambda^+_{\varepsilon}$};
  \foreach \x in {0,1,2,3,4}
  \draw[fill] (\x,0) circle (2pt) node[below] {$\lambda^0_{\varepsilon}$};
  \draw (0,0) -- (0,2pt) node[above] {$\lambda^+_{\varepsilon}$};
  \draw (0,0) -- (-2pt,0) node[below] {$\lambda^0_{\varepsilon}$};
  \draw (0,0) -- (2pt,0) node[below] {$\lambda^+_{\varepsilon}$};
  \draw (0,0) -- (4pt,0) node[below] {$\lambda^0_{\varepsilon}$};
  \draw (0,0) -- (6pt,0) node[below] {$\lambda^+_{\varepsilon}$};
  \draw (0,0) -- (-4pt,0) node[below] {$\lambda^0_{\varepsilon}$};
  \draw (0,0) -- (-6pt,0) node[below] {$\lambda^+_{\varepsilon}$};
  \draw[very thick] (0,0) -- (0,2pt) node[above] {$\lambda^+_{\varepsilon}$};
  \draw[very thick] (0,0) -- (-2pt,0) node[below] {$\lambda^0_{\varepsilon}$};
  \draw[very thick] (0,0) -- (2pt,0) node[below] {$\lambda^+_{\varepsilon}$};
  \draw[very thick] (0,0) -- (4pt,0) node[below] {$\lambda^0_{\varepsilon}$};
  \draw[very thick] (0,0) -- (6pt,0) node[below] {$\lambda^+_{\varepsilon}$};
  \draw[very thick] (0,0) -- (-4pt,0) node[below] {$\lambda^0_{\varepsilon}$};
  \draw[very thick] (0,0) -- (-6pt,0) node[below] {$\lambda^+_{\varepsilon}$};
  \draw (0,0) -- (0,2pt) node[above] {$\lambda^+_{\varepsilon}$};
  \draw (0,0) -- (-2pt,0) node[below] {$\lambda^0_{\varepsilon}$};
  \draw (0,0) -- (2pt,0) node[below] {$\lambda^+_{\varepsilon}$};
  \draw (0,0) -- (4pt,0) node[below] {$\lambda^0_{\varepsilon}$};
  \draw (0,0) -- (6pt,0) node[below] {$\lambda^+_{\varepsilon}$};
  \draw (0,0) -- (-4pt,0) node[below] {$\lambda^0_{\varepsilon}$};
  \draw (0,0) -- (-6pt,0) node[below] {$\lambda^+_{\varepsilon}$};
\end{tikzpicture}
\caption{Fig. 9}
\end{figure}

We remind here that $\lambda^+_{n\varepsilon}$ has multiplicity 2 and $\lambda^0_{n\varepsilon}$ has multiplicity one.
Let
\begin{equation}
C_k(r) = \{ z | z - 4 \alpha^2 k = r \} \quad k = 0, 1, 2, \ldots
\end{equation}
and
\begin{equation}
D_k = \{ z | z - 4 \alpha^2 k \leq r \} \quad k = 0, 1, 2, \ldots
\end{equation}
with \( r \leq \alpha^2 \), and let:

\begin{equation}
p^{(2)}_\varepsilon (D_k) = \frac{1}{2\pi i} \oint_{C_k(r)} (z - M^{(2)}_\varepsilon)^{-1} dz
\end{equation}
then

\begin{equation}
p^{(2)}_\varepsilon (D_0) = p^{(2)}_\varepsilon (\{ \lambda^+_\varepsilon \}) \quad \text{for } \varepsilon \text{ small enough}
\end{equation}

\begin{equation}
p^{(2)}_\varepsilon (D_k) = \begin{cases} 
p^{(2)}_\varepsilon (\{ \lambda^0_{k-1, \varepsilon} \}) & \text{k odd} \\
p^{(2)}_\varepsilon (\{ \lambda^0_{k-1, \varepsilon} \} \cup \{ \lambda^+_k \}) & \text{k even}
\end{cases}
\end{equation}
for \( \varepsilon \leq \varepsilon_k \) (see Fig. 9). We remark that \( \varepsilon_k \) cannot be chosen independent of \( k \).

**Theorem 5.2.** There exists \( \varepsilon_k > 0 \) such that for all \( z \in C_k(r) \) and all \( 0 < \varepsilon < \varepsilon_k \),

\begin{equation}
(z - M_l)^{-1} \text{ exists}
\end{equation}
Proof: It follows from Theorem 5.1 and the known properties of the spectrum of $M^{(Z)}_c$, rearranging the proof of Theorem 4.1 of Isaacson [2].

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

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

Moreover for all $\varepsilon$ sufficiently small $M_c$ possesses:

(i) two distinct eigenvalues $\lambda_1 (\varepsilon), \lambda_2 (\varepsilon) > 0$ such that:

$$\lim_{\varepsilon \to 0} \lambda_1 (\varepsilon) = \lambda_2 (\varepsilon) \equiv 0$$

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

$$\lim_{\varepsilon \to 0} \lambda_k (\varepsilon) = 4 \varepsilon^2 k \quad k = 1, 3, \ldots$$

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

$$\lim_{\varepsilon \to 0} \lambda_k (\varepsilon) = \lim_{\varepsilon \to 0} \lambda_k' (\varepsilon) = \lim_{\varepsilon \to 0} \lambda_k'' (\varepsilon) = 4 \varepsilon^2 k \quad k = 2, 4, \ldots$$

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

$$\sup_{z \in C_k (r)} \| (z - M_c)^{-1} - (z - M_c^{(Z)})^{-1} \| \leq \text{constant } r \frac{A_1}{T}$$
So that for \( \varepsilon \) sufficiently small

\[
\dim P_\varepsilon (D_k) = \dim P^{(2)}_\varepsilon (D_k)
\]

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 \( 0 < \lambda_0(\varepsilon) < \lambda_1(\varepsilon) < \lambda_2(\varepsilon) < \ldots \) be the eigenvalues of \( M_\varepsilon \). Then:

\[
\lim_{\varepsilon \to 0} \lambda_1(\varepsilon) = 0
\]

(5.21)

\[
\lim_{\varepsilon \to 0} \lambda_{2+n}(\varepsilon) = 4\alpha^2 (2n+1) \quad n = 0, 1, 2, \ldots
\]

(5.22)

\[
\lim_{\varepsilon \to 0} \lambda_{2+n}(\varepsilon) = \lim_{\varepsilon \to 0} \lambda_{2+n}(\varepsilon) = \lim_{\varepsilon \to 0} \lambda_{2+n}(\varepsilon) = 8\alpha^2 (n+1)
\]

\[
n = 0, 1, 2, \ldots
\]

(5.23)

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

\[
\lim_{\varepsilon \to 0} \| P_\varepsilon (S_k) - P^{(2)}_\varepsilon (S_k) \| = 0
\]

That is for \( \varepsilon \) sufficiently small

\[
\dim P_\varepsilon (S_k) = \dim P^{(2)}_\varepsilon (S_k)
\]

Theorem 5.4 follows now from Theorem 5.3.

A straightforward computation shows that the eigenvalues of \( N^{(2)}_\varepsilon \) when \( \varepsilon > 0 \) are given by:
where \( c = 2 ax_1 x_2 \), \( c_1 = 2 ax_1 (x_2 - x_1) < 0 \), \( c_2 = 2 ax_1 (x_2 - x_1) \) where \( x_1, x_2 \) are given in (i) of Proposition 2.2.

Let \( \{ -\overline{\lambda}_n \}_{n=0}^\infty \) 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 \( -\overline{\lambda}_n \leq -\overline{\lambda}_{n+1} \) \( n = 0, 1, \ldots \). Moreover if a number appears in more than one \( E_i \) \( i = 1, 2, 3 \) it will appear a corresponding number of times in \( \{ -\overline{\lambda}_n \}_{n=0}^\infty \) in particular since zero appears in \( E_1 \) and \( E_3 \) we will have \( -\overline{\lambda}_2 = -\overline{\lambda}_1 = 0 \).

**Theorem 5.5.** Let \( 0 \leq -\overline{\lambda}_0 (< \ldots < \overline{\lambda}_1 \ldots ) \) be the eigenvalues of \( N_\varepsilon \). Then

\[
\lim_{\varepsilon \to 0} -\overline{\lambda}_n (\varepsilon) = -\overline{\lambda}_n
\]

**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 \( \{ -\overline{\lambda}_n \}_{n=0}^\infty \) this corresponds to asymptotic eigenvalue degeneracy for \( N_\varepsilon \).

Since \( -\overline{\lambda}_2 = -\overline{\lambda}_1 = 0 \) we have

\[
\lim_{\varepsilon \to 0} -\overline{\lambda}_2 (\varepsilon) = -\overline{\lambda}_2 (\varepsilon) \equiv 0
\]
All the remaining \( \{-\overline{x_n}\}_{n=2}^\infty \) are distinct if \( \frac{x_1}{x_2} \) is irrational,

if \( \frac{x_1}{x_2} \) is rational \( \{-\overline{x_n}\}_{n=2}^\infty \) contains values that appear only once
and values that appear three times.

That is, there are eigenvalues of \( N_c \) that remain isolated when \( \epsilon \to 0 \) and eigenvalues that have asymptotic multiplicity three when \( \epsilon \to 0 \). We have already observed this phenomenon in the study of \( M_c \).
A NEW METHOD FOR GLOBAL OPTIMIZATION BASED ON STOCHASTIC DIFFERENTIAL EQUATIONS (U) CAMERINO UNIV (ITALY) MATHEMATICS INST F ALUFFI-PENTINI ET AL.
§6. The estimate of the first nonzero eigenvalue of $M_\varepsilon$ and $N_\varepsilon$.

In section 5 it has been shown that

$$(6.1) \quad -\lambda_0(\varepsilon) = -\lambda_0(\varepsilon) \equiv 0 \quad \forall \varepsilon \neq 0$$

$$(6.2) \quad \lim_{\varepsilon \to 0} -\lambda_1(\varepsilon) = \lim_{\varepsilon \to 0} -\lambda_1(\varepsilon) = 0$$

where $-\lambda_0(\varepsilon)$, $-\lambda_1(\varepsilon) > 0$ are the first two eigenvalues of $M_\varepsilon$ and $-\lambda_0(\varepsilon)$, $-\lambda_1(\varepsilon) > 0$ are the first two eigenvalues of $N_\varepsilon$.

In section 2 it has been shown that the eigenfunctions corresponding to $-\lambda_1(\varepsilon)$ and $-\lambda_2(\varepsilon)$ are respectively:

$$(6.3) \quad v_0(y) = \frac{d_0}{\sqrt{\varepsilon}} e^{-f_1y/2}$$

and

$$(6.4) \quad \bar{v}_0(y) = \frac{d_0}{\sqrt{\varepsilon}} e^{-f_2y/2}$$

where $f_1, f_2$ are given by (2.13), (2.14), $f_1c$, $f_2c$ by (2.4) and

$$(6.5) \quad d_\varepsilon = \left( \int_{-\infty}^{+\infty} e^{-f_1y/\varepsilon} dy \right)^{-1} = \frac{\varepsilon}{\sqrt{2}} c_\varepsilon$$

$$(6.6) \quad \bar{d}_\varepsilon = \left( \int_{-\infty}^{+\infty} e^{-f_2y/\varepsilon} dy \right)^{-1} = \frac{\varepsilon}{\sqrt{2}} \bar{c}_\varepsilon$$

are normalization constants such that $\|v_0\|_{L^2(\mathbb{R})} = \|ar{v}_0\|_{L^2(\mathbb{R})} = 1$ 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 \(-\lambda_2(\varepsilon) + \lambda_3(\varepsilon)\) and \(-\lambda_1(\varepsilon) + \lambda_0(\varepsilon)\) as \(\varepsilon \to 0\) that is the first nonzero eigenvalue of \(M_\varepsilon\) and \(N_\varepsilon\).

The same problem for the Fokker-Planck operators corresponding to \(M_\varepsilon\) and \(N_\varepsilon\) 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 \(-\lambda_2(\varepsilon), -\lambda_1(\varepsilon), M_\varepsilon\) be as above. Then as \(\varepsilon \to 0\) we have

\[
0 < -\lambda_1(\varepsilon) + \lambda_0(\varepsilon) \equiv -\lambda_1(\varepsilon) \leq \text{constant} \, e^{-\frac{2}{\varepsilon^2}} \lambda^4.
\]

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

\[
0 < -\lambda_1(\varepsilon) + \lambda_2(\varepsilon) \equiv -\lambda_1(\varepsilon) \leq \frac{\langle g, M \varepsilon g \rangle_{L^2(\mathbb{R})}}{\langle g, g \rangle_{L^2(\mathbb{R})}}
\]

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

Since \(v_0\) is an even function let us choose

\[
g = uv.
\]

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

\[
\frac{du}{dy} \in L^\infty(\mathbb{R}) \quad \text{where} \quad \frac{du}{dy} \quad \text{is the distributional derivative of} \quad u.
\]
The function $g$ is orthogonal to $v_0$ and belongs to the form domain of $M_\epsilon$.

We have

\[
< g, M_\epsilon g >_{L^2(\mathbb{R})} = \int_{-\infty}^{+\infty} u v_0 \left( -\frac{d^2}{dy^2} + V_\epsilon \right) u v_0 \, dy = \\
= \int_{-\infty}^{+\infty} \left\{ \left( \frac{d}{dy} (u v_0) \right)^2 + V_\epsilon u^2 v_0^2 \right\} \, dy = \\
= \int_{-\infty}^{+\infty} \left\{ \left( \frac{du}{dy} \right)^2 v_0^2 + u^2 \left( \frac{dv_0}{dy} \right)^2 + 2u \frac{du}{dy} v_0 \frac{dv_0}{dy} + V_\epsilon u^2 v_0^2 \right\} \, dy \\
= \int_{-\infty}^{+\infty} \left( \frac{du}{dy} \right)^2 v_0^2 \, dy
\]

Since

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

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

So that

\[
0 \leq -\lambda_1(\epsilon) + \lambda_0(\epsilon) \equiv -\lambda_1(\epsilon) \leq \frac{\int_{-\infty}^{+\infty} \left( \frac{du}{dy} \right)^2 v_0^2 \, dy}{\int_{-\infty}^{+\infty} u^2 v_0^2 \, dy}
\]
Let us choose

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

equation (6.12) becomes

\[ 0 < -\lambda_1(\varepsilon) + \lambda_0(\varepsilon) = -\lambda_1(\varepsilon) \leq \frac{\int_{-1}^{1} y^2 \, dy}{\int_{-\infty}^{+\infty} u^2 v_0^2 \, dy} \]

Moreover

\[ \int_{-1}^{1} y^2 \, dy = \frac{d}{\varepsilon} \int_{-1}^{+1} e^{-f_{1\varepsilon}(y)} \, dy = \frac{\varepsilon}{\sqrt{2}} c e \int_{-1}^{1} e^{-f_{1\varepsilon}(\varepsilon)} \, dy \]

\[ \leq \frac{\varepsilon}{\sqrt{2}} c e 2 e^{-f_{1\varepsilon}(1)} = \frac{\varepsilon}{\sqrt{2}} c e 2 e^{-2} e^{-\frac{\varepsilon^2}{2} - \frac{\varepsilon^2}{2}} \]

It can be easily shown that

\[ \lim_{\varepsilon \to 0} \varepsilon c \frac{\sqrt{2}}{\pi} \alpha \]

and that since \( x = \frac{\varepsilon}{\sqrt{2}} y \)
\[(6.16) \quad \lim_{\varepsilon \to 0} \int_{-\infty}^{\infty} v_\varepsilon^2 u^2 \, dy = 1 \]

In fact in the sense of distribution

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

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

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

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

**Theorem 6.2.** Let \(\overline{x}_2(\varepsilon), \overline{x}_1(\varepsilon), N_\varepsilon\) be as above. Then as \(\varepsilon \to 0\) we have

\[(6.18) \quad 0 < -\overline{x}_1(\varepsilon) + \overline{x}_0(\varepsilon) \equiv -\overline{x}_1(\varepsilon) \leq \text{constant} e^{-\frac{2}{\varepsilon^2}(f_2(x_1) - f_2(x_2))} \]

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

\[(6.19) \quad 0 < -\overline{x}_1(\varepsilon) + \overline{x}_0(\varepsilon) \equiv -\overline{x}_1(\varepsilon) \leq \frac{\int_{-\infty}^{\infty} \left( \frac{dh}{dy} \right)^2 v_0^2 \, dy}{\int_{-\infty}^{\infty} h^2 v_0^2 \, dy} \]
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

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

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

\[
(6.24) \quad \lim_{\varepsilon \to 0} \int_{-\infty}^{\infty} \bar{u}^2 \bar{v}_0^2 \, dy = 1
\]
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 \( e^{\frac{-x^2}{2}} f(x) \in L^1(\mathbb{R}) \) \( \forall \varepsilon \neq 0 \)

and suppose that

\[
f'(x) = 0
\]

has \( n \) roots \( \lambda_1, \lambda_2, \ldots, \lambda_n \) such that

\[
f''(\lambda_i) = \alpha_i \neq 0 \quad i = 1, 2, \ldots, n
\]

that is \( \lambda_1, \lambda_2, \ldots, \lambda_n \) are non-degenerate minimizers or maximizers of \( f \).

Let

\[
L_\varepsilon(\cdot) = \frac{\varepsilon^2}{2} \frac{\partial^2}{\partial x^2} + \frac{\varepsilon}{\partial x} \left( \frac{df}{dx} \right)
\]

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

\[
H_\varepsilon = -\frac{d^2}{dy^2} + W_\varepsilon(y)
\]

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

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

So that we expect the spectrum of \( H_\varepsilon \) to approximate the spectrum of \( n \) decoupled harmonic oscillators \( \frac{1}{k} = \frac{\lambda_i}{\alpha_i} \quad (2k+1) + \frac{\lambda_i}{\alpha_i} \),

\( i = 1, 2, \ldots n \) and \( k = 0, 1, 2, \ldots \).
In particular if $f$ has $m$ ($< n$) minimizers, that is $\alpha_{ij} > 0$

$j = 1, 2, \ldots, m$ we expect the eigenvalue zero of $H_{\varepsilon}$ (or $L_{\varepsilon}$) to have

asymptotically multiplicity $m$ when $\varepsilon \to 0$. 
References


[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.


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:

\[ f(x,y) = g(x,y) \ h(x,y) \]

with \( g(x,y) = 1 + x^2 (16 - 30x + 15x^2) \)

\[ h(x,y) = 30 + x^2 (16 - 18x + 3x^2) \]

where \( u = x - 1, \) and \( v = 2x - 1 \)
Problem 11
A function with three ill-conditioned minima, \( a = 100 \)

a) function: as in probl. 10
b) parameter values: \( a = 100, \ b = 1/a \)
c), d), e): as in problem 10
f) minima in the region \( D \): two global minima at

\[
(x, y) = (0, 2.60891), \ \text{where} \ f = -18.05870
\]
and another local minimum at

\[
(x, y) = (0, 0), \ \text{where} \ f = 0
\]
g), h), i): as in problem 10.

Problem 12
A function with three ill-conditioned minima, \( a = 1000 \)

a) function: as in probl. 10
b) parameter values: \( a = 1000, \ b = 1/a \)
c), d), e): as in problem 10
f) minima in the region \( D \): two global minima at

\[
(x, y) = (0, 4.70174), \ \text{where} \ f = -227.76575
\]
and another local minimum at

\[
(x, y) = (0, 0), \ \text{where} \ f = 0
\]
g), h), i): as in problem 10.

Problem 13
A function with three ill-conditioned minima, \( a = 10000 \)

a) function: as in probl. 10
b) parameter values: \( a = 10000, \ b = 1/a \)
c), d), e): as in problem 10
f) minima in the region \( D \): two global minima at

\[
(x, y) = (0, 8.37401), \ \text{where} \ f = -2429.41477
\]
and another local minimum at

\[
(x, y) = (0, 0), \ \text{where} \ f = 0
\]
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,
   leaving 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: $D = \{ |x| \leq 10, \quad |y| \leq 10 \}$
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
\[
\begin{align*}
(x,y) &= (-7.08350, -7.70831) \\
(x,y) &= (-0.80032, -7.70831) \\
(x,y) &= (5.48286, -7.70831) \\
(x,y) &= (-7.70831, -0.80032) \\
(x,y) &= (-1.42513, -7.08350) \\
(x,y) &= (4.85805, -7.08350) \\
(x,y) &= (-7.08350, -1.42513) \\
(x,y) &= (-0.80032, -1.42513) \\
(x,y) &= (5.48286, -1.42513) \\
(x,y) &= (-7.70831, -0.80032) \\
(x,y) &= (-1.42513, -0.80032) \\
(x,y) &= (4.85805, -0.80032) \\
(x,y) &= (7.08350, 4.85805) \\
(x,y) &= (-8.00320, 4.85805) \\
(x,y) &= (5.48286, 4.85805) \\
(x,y) &= (-7.70831, 5.48286) \\
(x,y) &= (-1.42513, 5.48286) \\
(x,y) &= (4.85805, 5.48286), \text{ where } f = -186.73091 \\
\end{align*}
\]
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 $g(x,y)$ 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:
\[ (x, y) = t(2.98978, 0), \text{ where } f = 0.46981 \]
\[ (x, y) = t(5.96370, 0), \text{ where } f = 1.97693 \]
\[ (x, y) = t(8.87846, 0), \text{ where } f = 4.21128 \]
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:
\[ f(x, y) = (4 - 2.1 x^2 + x^4/3) x^2 + xy + (-4 + 4 y^2) y^2 \]

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
\[ (x, y) = t(-0.089842, 0.71266), \text{ where } f = -1.0316 \]
and four other local minima, at
\[ (x, y) = t(-1.70361, 0.79608), \text{ where } f = -0.21546, \text{ and} \]
\[ (x, y) = t(1.60710, 0.56865), \text{ where } f = 2.10425 \]
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
\[
\begin{align*}
x &= -7.70831, -1.42513, 4.85806, \quad \text{where } f &= -12.87088 \\
\end{align*}
\]
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 \( 2\pi \)).

Problem 4
A fourth-order polynomial in two variables

a) function:
\[
f(x) = x^4/4 - x^2/2 + ax + y^2/2
\]

b) parameter values: \( a = 0.1 \)
c) dimension: \( N = 2 \)
d) region: \( D = (|x| \leq 10, |y| \leq 10) \)
e) penalization: none
f) minima in the region \( D \):
\begin{itemize}
  \item two minima, both for \( y = 0 \), as in problem 1
\end{itemize}
g) initial point: \( (x_2, y_0) = (1, 0) \)
h) source: as in problem 1.
APPENDIX I. 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
\[ x = -1.04668, \text{ where } f = -0.35239, \]
and another local minimum at
\[ x = 0.94565, \text{ where } f = -0.15264 \]
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
\[ x = \pm 3, \text{ where } f = 7, \]
and another local minimum at
\[ x = 0, \text{ where } f = 250 \]
g) initial point: \( x_0 = 0 \)
h) source: ref. 6 of [15]
i) notes: the starting point is exactly at the non-global minimizer, midway between the global ones.
References


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 B 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 optimization 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.
J. 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. [6]). The FORTRAN implicit type definition for integers is used throughout; all non-integer variables are double-precision.

The call statement of GLOMIP is:

\[
\text{CALL GLOMIP (NPROB, N, XO, XMIN, XMAX)}
\]

where

- \( \text{NPROB} \) is the (input) number which identifies the test problem according to the sequence in Appendix I.
- \( \text{N} \) is the (output) dimension of the problem (number of independent variables).
- \( \text{XO} \) is the (output) \( N \)-vector containing the suggested initial point \( x_0 \).
- \( \text{XMIN} \) and \( \text{XMAX} \) are (output) \( N \)-vectors containing the boundaries of the observation region \( D \), defined by
\[
\text{XMIN}(I) \leq \text{X}(I) \leq \text{XMAX}(I), \quad I = 1, \ldots, N
\]

The call statement of GLOMTF is:

\[
\text{CALL GLOMTF (NPROB, N, X, FUNZ)}
\]

where

- \( \text{NPROB} \) is the (input) problem number (see above).
- \( \text{N} \) is the (input) problem dimension (must be equal to the value provided by GLOMIP).
- \( \text{X} \) is the (input) \( N \)-vector containing the point \( x \) at which the test function is to be computed.
- \( \text{FUNZ} \) is the (output) value of the (possibly penalized) test function at \( x \).
Therefore a test problem, such as we provide 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 $\mathbb{R}^N$, 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

$$D = \{ x_{\text{min}_i} \leq x_i \leq x_{\text{max}_i} , i = 1, \ldots, N \}$$

e) penalization function $w$ (if any), which is always defined by means of a standard penalization function $u$ of a single real variable $x$ (with $a > 0$, $b > 0$)

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

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 $\mathbb{R}^N$ such that $f(x) \geq f(x^*)$ for all $x$ in $\mathbb{R}^N$.

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 [51]), 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 $x_0$. 
I. 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.
Problem 17
Penalized Branin function

a) function:
\[ f(x,y) = (y - bx^2 + cx - 6)^2 + 10 \left( 1 - \frac{1}{f} \cos x \right) + 1 \]

b) parameter values: \( b = 5.1/(4\pi^2), \ c = 5/\pi, \ f = 1/(8\pi) \)

c) dimension: \( N = 2 \)

d) region: \( D = \{(x, y) : -2 \leq x \leq 4, \ 0 \leq y \leq 15 \} \)

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 \( f = 50 \)
  \( (x,y) = (-1.8, 2) \), 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 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

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

b) parameter values: $M = 5$

c) dimension: $N = 4$

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

e) penalization:

$$w(x) = \sum_{j=1}^{N} u(x - 5, 5, 100, 2)$$
f) minima in the region $D$ : a global minimum at
$$x = (4.00004, 4.00013, 4.00004, 4.00013), \text{ where } f = -10.15320$$
and four other local minima, at
$$x = (1.00013, 1.00016, 1.00013, 1.00016), \text{ where } f = -5.05520$$
$$x = (3.00180, 6.99833, 3.00180, 6.99833), \text{ where } f = -2.63047$$
$$x = (5.99875, 6.00029, 5.99875, 6.00029), \text{ where } f = -2.68286$$
$$x = (7.99958, 7.99964, 7.99958, 7.99964), \text{ where } f = -5.10077$$

g) initial point: $x_{0j} = 9$, $j = 1, \ldots, N$

h) source: Appendix I 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

d) minima in the region $D$ : a global minimum at
$$x = (4.00057, 4.00069, 3.99949, 3.99961), \text{ where } f = -10.40294$$
and six other local minima, at
$$x = (1.00023, 1.00027, 1.00018, 1.00022), \text{ where } f = -5.08767$$
$$x = (2.00481, 8.99168, 2.00462, 9.99150), \text{ where } f = -1.85759$$
$$x = (3.00091, 7.00064, 3.00027, 7.00010), \text{ where } f = -2.76590$$
$$x = (4.99423, 4.99500, 3.00606, 3.00683), \text{ where } f = -3.72450$$
$$x = (5.99811, 6.00008, 5.99733, 5.99931), \text{ where } f = -2.75193$$
$$x = (7.99951, 7.99962, 7.99950, 7.99961), \text{ where } f = -5.10077$$

q)., n), 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:

$$f(x) = -\sum_{i=1}^{M} c_i \exp \left[ -\sum_{j=1}^{N} a_{ij} (x - p_{ij})^2 \right]$$

where

$$c = \begin{bmatrix} c_1 \\ \vdots \end{bmatrix} = \begin{bmatrix} 1.1 & 1.2 & 5 & 3.2 \end{bmatrix}^T$$
b) parameter values: $M = 4$

\[
A = \begin{bmatrix}
3 & 10 & 30 \\
0.1 & 10 & 35 \\
3 & 10 & 30 \\
0.1 & 10 & 35 \\
\end{bmatrix}
\]

\[
P = \begin{bmatrix}
0.3689 & 0.1170 & 0.2673 \\
0.4699 & 0.4387 & 0.7470 \\
0.1091 & 0.8732 & 0.5547 \\
0.03815 & 0.5743 & 0.8828 \\
\end{bmatrix}
\]

c) dimension: $N = 3$

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

e) penalization:

\[
w(x) = \sum_{j=1}^{N} u(x - 0.5, 0.5, 100, 2)
\]

f) minima in the region $D$: a global minimum at

\[
x = (0.11461, 0.55565, 0.85255), \ \text{where} \ \ f = -3.86278
\]

and two other local minima at

\[
x = (0.10934, 0.86052, 0.56412), \ \text{where} \ \ f = -3.08976
\]

\[
x = (0.36872, 0.11758, 0.26757), \ \text{where} \ \ f = -1.00082
\]

g) initial point: \ $x_{0j} = 0.5, \ j = 1, \ldots, N$

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

i) notes: none.
Problem 22
Penalized six-dimensional Hartman function, $N = 6$

a) function: as in problem 21
b) parameter values: $M = 4$

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

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

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

\[
w(x) = \sum_{j=1}^{N} u(x - 0.5, 0.5, 100, 2)
\]
f) minima in the region $D$: a global minimum at

\[
x = (0.20169, 0.15001, 0.47687, 0.27533, 0.31165, 0.65730)
\]
where $f = -3.322137$, 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: $x_{0j} = 0.5, \ j = 1, \ldots, N$
h) source: Appendix 1 in ref. [3]
i) notes: none.
Problem 23
Penalized Levy-Montalvo function, type 1, N = 2

a) function:
\[ f(x) = \left( \frac{\pi}{N} \right) \left( 10 \sin^2(\pi y_1) + (y_N - 1)^2 + \sum_{j=1}^{N-1} (y_j - 1)^2 \left[ 1 + 10 \sin^2(\pi y_{j+1}) \right] \right) \]
where \( y_j = 1 + (x_j - 1)/4 \), \( j = 1, \ldots, N \)

b) parameter values: none
c) dimension: \( N = 2 \)
d) region: \( D = \{ -10 \leq x_j \leq 10, \ j = 1, \ldots, N \} \)
e) penalization:
\[ w(x) = \sum_{j=1}^{N} u(x_j, 10, 100, 4) \]
f) minima in the region \( D \): a single global minimum at \( x_j = 1, \ j = 1, \ldots, N \), where \( f = 0 \)
and a number of local minima of the order of \( 5^N \)
g) initial point: \( x_{0j} = 0, \ j = 1, \ldots, N \)
h) source: ref. [5]
i) notes: three-dimensional plots of \( f \) are given in [5].

Problem 24
Penalized Levy-Montalvo function, type 1, N = 3

a), b): as in problem 23
c) dimension: \( N = 3 \)
d), e), f), g), h), i): as in problem 23.
Problem 25
Penalized Levy-Montalvo function, type 1, \( N = 4 \)

a), b) : as in problem 23

c) dimension: \( N = 4 \)

d), e), f), g), h), i) : as in problem 23.

Problem 26
Penalized Levy-Montalvo function, type 2, \( N = 5 \)

a) function: as in problem 23, but with

\[ y_i = x_i, \quad i = 1, \ldots, N \]

b) : as in problem 23

c) dimension: \( N = 5 \)

d), e) : as in problem 23

f) minima in the region \( D \):

- a single global minimum as in problem 23,
- and a number of local minima of the order of \( 10^N \)

g), h), i) : as in problem 23.

Problem 27
Penalized Levy-Montalvo function, type 2, \( N = 8 \)

a), b) : as in problem 26

c) dimension: \( N = 8 \)

d), e), f), g), h), i) : as in problem 26.
Problem 28
Penalized Levy-Montalvo function, type 2, \( N = 10 \)

a), b) : as in problem 26

c) dimension: \( N = 10 \)

d), e), f), g), h), i) : as in problem 26.

Problem 29
Penalized Levy-Montalvo function, type 3, range 10, \( N = 2 \)

a) function:

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

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

\[x_j = 1, \quad j = 1, \ldots, N, \text{ where } f = 0\]

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: \( D = (-5 \leq x_i \leq 5, \ i = 1,...,N) \)

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:
\[
f(x) = \left( \sum_{j=1}^{5} j \cdot x_j \right)^{2/4}
\]

b) parameter values: none

c) dimension: \( N = 5 \)

d) region: \( D = (-20000 \leq x_j \leq 10000, \; j = 1, \ldots, N) \)

e) penalization: none

f) minima in the region \( D \): a single global minimum at
\[ x_j = 0, \; j = 1, \ldots, N, \; \text{where } f = 0 \]

g) initial point: \( x_{0j} = 1000, \; j = 1, \ldots, N \)

h) source: suggested by one of the authors (V.P.)

i) notps: 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
\[
g(x) = \exp \left( -\frac{S}{b^2 - S} \right) (S < b^2)
\]
\[
= 0 (S \geq b^2)
\]
\[
S = \|x - c\|^2, \quad C = 100\|c\|^2
\]

and
\[
x = (x_1, x_2, \ldots, x_N), \quad c = (a, 0, 0, \ldots, 0)
\]

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

c) dimension: \( N = 2 \)

d) region:
\[
D = \{-1000 \leq x_j \leq 1000, \ j = 1, \ldots, N\}
\]

e) penalization: none

f) minima in the region \( D \): a single global minimum at \( x = (99.99001, 0) \), where \( f = -10.99885 \)

and another local minimum at \( x = (0, 0) \), where \( f = 0 \)

g) initial point: \( x_0 = (0, 100) \)

h) source: suggested by one of the authors (F.A.P.)

i) notes: the perturbation to \( \|x\|^2 \), which contains the global minimum, is "visible" only in a small neighborhood of the point \( c \).
Problem 37
A function with a global minimum having a small region of attraction, $a = 100$, $N = 5$

a) functions as in problem 36
b) parameter values: $a = 10$, $b = 1$, $h = 10$
c) dimension: $N = 5$
d) region: $D = \{-100 \leq x_j \leq 100, \ j = 1, \ldots, N\}$
e) penalization: none
f) minima in the region $D$: a single global minimum at
   $x = (9.91026, 0, 0, 0, 0)$, where $f = -10.89732$
   and another local minimum at
   $x = (0, 0, 0, 0, 0)$, where $f = 0$
g) initial point: $x_0 = (0, 0, 0, 0, 10)$
h) i) as in probl. 36.
APPENDIX 2

The FORTRAN listing
A FUNCTION WITH A SMALL-ATTACHMENT-REGION GLOBAL MINIMUM (N = 2)

```
C 36 CONTINUE

N = 2
X(1) = 0.00
X(2) = 100.00
VMIN = -1000.00
VMAX = 1000.00
GO TO 700
```

C A FUNCTION WITH A SMALL-ATTACHMENT-REGION GLOBAL MINIMUM (N = 5)

```
C 37 CONTINUE

N = 5
X(1) = 0.00
X(2) = 0.00
X(3) = 0.00
X(4) = 0.00
X(5) = 10.00
VMIN = -100.00
VMAX = 100.00
GO TO 700
```

C CONTINUE

```
DU 773 I = 1,N
XMIN(I) = VMIN
XMAX(I) = VMAX
```

C CONTINUE

```
RETURN
```

C CONTINUE

```
DU I1 J = 1,N
XMIN(J) = VMIN
XMAX(J) = VMAX
```

C CONTINUE

```
RETURN
```

C CONTINUE

```
RETURN
```

C CONTINUE

```
RETURN
```

C CONTINUE

```
RETURN
```

C CONTINUE

```
RETURN
```

C CONTINUE

```
RETURN
```
GO TO 315
C 28 PENALIZED LEVY-MONTALVO FUNCTION, TYPE 2 (N = 10)
C 280 CONTINUE
N = 10
GO TO 315
C 29 PENALIZED LEVY-MONTALVO FUNCTION, TYPE 3, RANGE = 10 (N = 2)
C 290 CONTINUE
N = 2
GO TO 315
C 30 PENALIZED LEVY-MONTALVO FUNCTION, TYPE 3, RANGE = 10 (N = 3)
C 310 CONTINUE
N = 3
GO TO 315
C 31 PENALIZED LEVY-MONTALVO FUNCTION, TYPE 3, RANGE = 10 (N = 4)
C 315 CONTINUE
N = 4
C 32 PENALIZED LEVY-MONTALVO FUNCTION, TYPE 3, RANGE = 5 (N = 5)
C 320 CONTINUE
N = 5
GO TO 345
C 33 PENALIZED LEVY-MONTALVO FUNCTION, TYPE 3, RANGE = 5 (N = 6)
C 330 CONTINUE
N = 6
GO TO 345
C 34 PENALIZED LEVY-MONTALVO FUNCTION, TYPE 3, RANGE = 5 (N = 7)
C 340 CONTINUE
N = 7
C 345 CONTINUE
N = 7
C 35 PENALIZED LEVY-MONTALVO FUNCTION WITH A SINGLE CUBIC-SHAPED MINIMUM (N = 5)
C 36 CONTINUE
C 19 PENALIZED SHEKEL FUNCTION, M = 7 (N = 4)

C 20 PENALIZED SHEKEL FUNCTION, M = 10 (N = 4)

C 21 PENALIZED THREE-DIMENSIONAL HARTMAN FUNCTION (M = 3)

C 22 PENALIZED SIX-DIMENSIONAL HARTMAN FUNCTION (M = 6)

C 23 PENALIZED LEVY-MONTALVO FUNCTION, TYPE 1 (N = 2)

C 24 PENALIZED LEVY-MONTALVO FUNCTION, TYPE 1 (N = 3)

C 25 PENALIZED LEVY-MONTALVO FUNCTION, TYPE 1 (N = 4)

C 26 PENALIZED LEVY-MONTALVO FUNCTION, TYPE 2 (N = 5)

C 27 PENALIZED LEVY-MONTALVO FUNCTION, TYPE 2 (N = 8)

C 157 CONTINUE

C 260 CONTINUE

C 210 CONTINUE

C 223 CONTINUE

C 245 CONTINUE

C 257 CONTINUE

C 271 CONTINUE

C 293 CONTINUE
c 15 a function with three ill-conditioned minima, a=10 (n = 2)
 150 continue
 11 a function with three ill-conditioned minima, a=10**2 (n = 2)
 110 continue
 12 a function with three ill-conditioned minima, a=10**3 (n = 2)
 120 continue
 13 a function with three ill-conditioned minima, a=10**4 (n = 2)
 130 continue
 14 a function with three ill-conditioned minima, a=10**5 (n = 2)
 140 continue
 15 a function with three ill-conditioned minima, a=10**6 (n = 2)
 150 continue
    q = 2
    x0(1) = 0.00
    x0(2) = 0.00
    xmin(1) = -10.00
    xmin(2) = -10.00
    xmax(1) = 10.00
    xmax(2) = 10.00
    return

c 10 goldstein-price function (n = 2)
 160 continue
    q = 2
    v0 = 1.00
    vmin = -2.00
    vmax = 2.00
    go to 160

c 17 penalized bramin function (n = 2)
 170 continue
    q = 2
    x0(1) = 2.00
    x0(2) = 2.00
    xmin(1) = -2.00
    xmin(2) = -2.00
    xmax(1) = 2.00
    xmax(2) = 2.00
    return

1 penalized chebel function, m = 2 (m = 4)
 1 continue
A FOURTH ORDER POLYNOMIAL IN TWO VARIABLES (n = 2)

\[ \text{CONTINUE} \]

\[ \chi = 2 \]

\[ v_0 = 3.00 \]

\[ v_{\min} = -1.00 \]

\[ v_{\max} = 10.00 \]

\[ \text{GO TO 100} \]

A FUNCTION WITH A SINGLE ROW OF LOCAL MINIMA (n = 2)

\[ \text{CONTINUE} \]

\[ \chi = 2 \]

\[ v_0(1) = 1.00 \]

\[ v_0(2) = 0.00 \]

\[ v_{\min} = -10.00 \]

\[ v_{\max} = 10.00 \]

\[ \text{GO TO 100} \]

SIX-HUMP CAMEL FUNCTION (n = 2)

\[ \text{CONTINUE} \]

\[ \chi = 2 \]

\[ v_0 = 3.00 \]

\[ v_{\min}(1) = -3.00 \]

\[ v_{\max}(1) = -15.00 \]

\[ v_{\min}(2) = -5.00 \]

\[ v_{\max}(2) = 15.00 \]

\[ \text{RETURN} \]

TWO-DIMENSIONAL PENALIZED SHUBERT FUNCTION, BETA = 0 (n = 2)

\[ \text{CONTINUE} \]

TWO-DIMENSIONAL PENALIZED SHUBERT FUNCTION, BETA = 1/2 (n = 2)

\[ \text{CONTINUE} \]

TWO-DIMENSIONAL PENALIZED SHUBERT FUNCTION, BETA = 1 (n = 2)

\[ \text{CONTINUE} \]

\[ \chi = 2 \]

\[ v_{\min} = -10.00 \]

\[ v_{\max} = 10.00 \]

\[ \theta = 3.00 \]
SUBROUTINE GLORIP (NPROB, N, XIN, XMAX)

THE SUBROUTINE GLORIP PROVIDES THE CODING FOR THE NUMBER
OF VARIABLES, THE INITIAL POINT, AND THE OBSERVATION REGION
TO BE USED, TOGETHER WITH THE 17 TEST FUNCTIONS GIVEN BY
SUBROUTINE GLORIF, TO DEFINE 37 TEST PROBLEMS FOR GLOBAL
MINIMIZATION SOFTWARE.

THE SUBROUTINE GLORIP RETURNS IN N, X0, AND XIN, XMAX
THE NUMBER OF VARIABLES, THE INITIAL POINT, AND THE
BOUNDARIES OF THE OBSERVATION REGION.

CALLING STATEMENT
CALL GLORIP (NPROB, N, XIN, XMAX)

DESCRIPTION OF THE CALL PARAMETERS
(THE FORTRAN IMPLICIT TYPE DEFINITION FOR INTEGERS IS USED.
ALL NON-INTEGER PARAMETERS ARE DOUBLE-PRECISION).

NPROB IS THE (INPUT) NUMBER OF THE TEST PROBLEM TO BE
CONSIDERED.
N IS THE (OUTPUT) NUMBER OF VARIABLES (DIMENSION) OF
THE PROBLEM.
XMIN, XMAX ARE THE (OUTPUT) N-VECTORS CONTAINING THE LEFT
AND RIGHT BOUNDARIES OF THE OBSERVATION REGION
DEFINED BY THE POINTS X = (X1,...,XN) SUCH THAT
XMIN(I) .LE. X(I) & XMAX(I), I = 1,...,N.

DOUBLE PRECISION X0, XMIN, XMAX
DOUBLE PRECISION V0, VMIN, VMAX

DIMENSION X0(N), XMIN(N), XMAX(N)

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

1 CONTINUE
M = 1
V0 = 1.0
VMIN = -10.0
VMAX = 10.0
GO TO 300

2 CONTINUE
M = 1
V0 = 1.0
VMIN = -10.0
VMAX = 10.0
GO TO 300

3 CONTINUE
M = 1
V0 = 1.0
VMIN = -10.0
VMAX = 10.0
GO TO 300

1 A FOURTH-ORDER POLYNOMIAL (N = 1)
N = 1
V0 = 1.0
VMIN = -10.0
VMAX = 10.0
GO TO 300

2 GOLDSMITH SIXTH ORDER POLYNOMIAL (N = 1)
N = 1
V0 = 1.0
VMIN = -4.0
VMAX = 4.0
GO TO 300

3 ONE-DIMENSIONAL PENALTY FUNCTION (N = 1)
C 75 A FUNCTION WITH A SMALL-ATTRACTION-REGION GLOBAL MINIMUM (N = 2)
C 520 CONTINUE
   P6 = P76A
   RP = P76B
   M = P76C
   PUND = P76D
   GOTO 373
C 77 A FUNCTION WITH A SMALL-ATTRACTION-REGION GLOBAL MINIMUM (N = 5)
C 577 CONTINUE
   PG = P77A
   RP = P77B
   M = P77C
   PUND = P77D
   577 CONTINUE
   S = 7.90
   DO 377 1 = 2, M
       S = S + X(I) * X(I)
   377 CONTINUE
   FUNZ = S + X(I) * X(I)
   S = S + (X(I) - RG)**2
   IF (S + X(RP*RP*PUND) ) FUNZ = FUNZ - (RG*RG*M)**DEXP(-S/(RP*RP-S))
RETURN
C END
34

3.3 CONTINUE

C 3.4 CONTINUE

C 3.5 CONTINUE

C 3.6 CONTINUE

C 3.7 CONTINUE

C 3.8 CONTINUE

C 3.9 CONTINUE

C 3.10 CONTINUE

C 3.11 CONTINUE

C 3.12 CONTINUE

C 3.13 CONTINUE

C 3.14 CONTINUE

C 3.15 CONTINUE

C 3.16 CONTINUE

C 3.17 CONTINUE

C 3.18 CONTINUE

C 3.19 CONTINUE

C 3.20 CONTINUE

C 3.21 CONTINUE

C 3.22 CONTINUE

C 3.23 CONTINUE

C 3.24 CONTINUE

C 3.25 CONTINUE

C 3.26 CONTINUE

C 3.27 CONTINUE

C 3.28 CONTINUE

C 3.29 CONTINUE

C 3.30 CONTINUE

C 3.31 CONTINUE

C 3.32 CONTINUE

C 3.33 CONTINUE

C 3.34 CONTINUE

C 3.35 CONTINUE

C 3.36 CONTINUE

C 3.37 CONTINUE

C 3.38 CONTINUE

C 3.39 CONTINUE

C 3.40 CONTINUE

C 3.41 CONTINUE

C 3.42 CONTINUE

C 3.43 CONTINUE

C 3.44 CONTINUE

C 3.45 CONTINUE

C 3.46 CONTINUE

C 3.47 CONTINUE

C 3.48 CONTINUE

C 3.49 CONTINUE

C 3.50 CONTINUE

C 3.51 CONTINUE

C 3.52 CONTINUE

C 3.53 CONTINUE

C 3.54 CONTINUE

C 3.55 CONTINUE

C 3.56 CONTINUE

C 3.57 CONTINUE

C 3.58 CONTINUE

C 3.59 CONTINUE

C 3.60 CONTINUE

C 3.61 CONTINUE

C 3.62 CONTINUE

C 3.63 CONTINUE

C 3.64 CONTINUE

C 3.65 CONTINUE

C 3.66 CONTINUE

C 3.67 CONTINUE

C 3.68 CONTINUE

C 3.69 CONTINUE

C 3.70 CONTINUE

C 3.71 CONTINUE

C 3.72 CONTINUE

C 3.73 CONTINUE

C 3.74 CONTINUE

C 3.75 CONTINUE

C 3.76 CONTINUE

C 3.77 CONTINUE

C 3.78 CONTINUE

C 3.79 CONTINUE

C 3.80 CONTINUE

C 3.81 CONTINUE

C 3.82 CONTINUE

C 3.83 CONTINUE

C 3.84 CONTINUE

C 3.85 CONTINUE

C 3.86 CONTINUE

C 3.87 CONTINUE

C 3.88 CONTINUE

C 3.89 CONTINUE

C 3.90 CONTINUE

C 3.91 CONTINUE

C 3.92 CONTINUE

C 3.93 CONTINUE

C 3.94 CONTINUE

C 3.95 CONTINUE

C 3.96 CONTINUE

C 3.97 CONTINUE

C 3.98 CONTINUE

C 3.99 CONTINUE

C 4.0 CONTINUE

C 4.1 CONTINUE

C 4.2 CONTINUE

C 4.3 CONTINUE

C 4.4 CONTINUE

C 4.5 CONTINUE

C 4.6 CONTINUE

C 4.7 CONTINUE

C 4.8 CONTINUE

C 4.9 CONTINUE

C 5.0 CONTINUE

C 5.1 CONTINUE

C 5.2 CONTINUE

C 5.3 CONTINUE

C 5.4 CONTINUE

C 5.5 CONTINUE

C 5.6 CONTINUE

C 5.7 CONTINUE

C 5.8 CONTINUE

C 5.9 CONTINUE
& = 4
FNN = 0.000
\( y_{1, J} = 1.00 \)
CJ = 1.00
YL = 2.00
S = S^P.L1A(1,J) \times (y_{1,J})^2 \times P.2b(1,J)^2
1 \text{ CONTINUE}
\text{IF} \ (G \geq P.2c) \ FNN = FUN1-P.2c(1) \times 0.E(5)
217 \text{ CONTINUE}
GJ = 1.00
C
24 \text{ PENALIZED SIX-DIMENSIONAL HARTREAN FUNCTION \( (N = 6) \)}
C
215 \text{ CONTINUE}
N = 4
FNN = 0.000
\( y_{\ldots, 1} = 1.00 \)
CJ = 1.00
YL = 2.00
S = S^P.L1A(1,J) \times (y_{\ldots, 1})^2 \times P.2b(1,J)^2
219 \text{ CONTINUE}
\text{IF} \ (G \geq P.2c) \ FNN = FUN1-P.2c(1) \times 0.E(5)
C
26 \text{ PENALIZED LEVY-MONTALVO FUNCTION, TYPE 1 \( (N = 2) \)}
C
29 \text{ CONTINUE}
C
24 \text{ PENALIZED LEVY-MONTALVO FUNCTION, TYPE 1 \( (N = 3) \)}
C
26 \text{ CONTINUE}
C
25 \text{ PENALIZED LEVY-MONTALVO FUNCTION, TYPE 1 \( (N = 4) \)}
C
219 \text{ CONTINUE}
\( y_{1, 1} = 1.00 \)
\( (y_{1,1} = 1.00) \times 0.2500 \times (4(1)-1.00) \)
25 \text{ CONTINUE}
GJ = 2.00
C
26 \text{ PENALIZED LEVY-MONTALVO FUNCTION, TYPE 2 \( (N = 5) \)}
C
29 \text{ CONTINUE}
C
27 \text{ PENALIZED LEVY-MONTALVO FUNCTION, TYPE 2 \( (N = 6) \)}
C
29 \text{ CONTINUE}
C
28 \text{ PENALIZED LEVY-MONTALVO FUNCTION, TYPE 2 \( (N = 10) \)}
C
\text{CONTINUE}
C 10 SOLSTEEL-PRICE FUNCTION (N = 2)
C
10 CONTINUE
U = X(1)*X(1)+1.0
V = 3.0+X(1)^2-1.0+X(2)
UU = U*U
VV = V*V
F0U = (5.0*U+UU*(16.0-0.0-0.0-0.0+3.0*UU))
1 = (3.0*V+VV*(19.0-16.0+V+3.0*VV))
RETURN
C 17 PENALIZED GRANIX FUNCTION (N = 2)
C
17 CONTINUE
FUNZ = (X(2)-P17A*(X(1)/P1))^4+(5.0/P1)*X(1)-6.0)**2
1+(10.0*(1.0)*1.0/(1.0+P1))*SIN(X(1))+(10.0)
IF (DABS(X(2)-1.05)*5.00+0.00+2.0)
IF (DABS(X(2)-1.05)*5.00+0.00+2.0)
1 FUN1 = FUN4*PENFUN(X(2)-1.05,0.50,100.00,2)
IF (DABS(X(2)-1.05)*5.00+0.00+2.0)
1 FUN2 = FUN3*PENFUN(X(2)-1.05,0.50,100.00,2) RETURN
C 16 PENALIZED SHEKEL FUNCTION, N = 5 (N = 4)
C
16 CONTINUE
N = 5
60 Go To 203
C 19 PENALIZED SHEKEL FUNCTION, N = 7 (N = 4)
C
19 CONTINUE
N = 7
60 Go To 203
C 20 PENALIZED SHEKEL FUNCTION, N = 10 (N = 4)
C
20 CONTINUE
N = 10
C 23 CONTINUE
FUNZ = 0.0
DO 20 J = 1,N
J = P20(J)
DO 20 J = 1,N
S = S+(X(J)-P20(A(J)))**2
20 CONTINUE
FUNZ = FUNZ + 1.0/25
C 27 CONTINUE
C 27 CONTINUE
1 = (1.0)+(DABS(X(1)-1.0))*5.00
1 = (1.0)+(DABS(X(1)-1.0))*5.00
CONTINUE
RETURN
C PENALIZED THREE-DIMENSIONAL HARTMAN FUNCTION (N = 3)
C
1 CONTINUE
31

S2 = SL*DI-OGOS((DI+1.00)*A(2)+DI)
S5 CONTINUE
FUNZ = FUNZ+51.52
IF (5ARC(X(1))>10.00) FUNZ = FUNZ+PENFUN(X(1),10.00,100.00,2)
IF (5ARC(X(2))>10.00) FUNZ = FUNZ+PENFUN(X(2),10.00,100.00,2)
RETURN
C 10 A FUNCTION WITH THREE ILL-CONDITIONED MINIMA, A=10 (N = 2)
C 10 CONTINUE
A = 1.01
B = 1.0-1
GO TO 155
C 11 A FUNCTION WITH THREE ILL-CONDITIONED MINIMA, A=10**2 (N = 2)
C 11 CONTINUE
A = 1.02
B = 1.0-2
GO TO 155
C 12 A FUNCTION WITH THREE ILL-CONDITIONED MINIMA, A=10**3 (N = 2)
C 12 CONTINUE
A = 1.03
B = 1.0-3
GO TO 155
C 13 A FUNCTION WITH THREE ILL-CONDITIONED MINIMA, A=10**4 (N = 2)
C 13 CONTINUE
A = 1.04
B = 1.0-4
GO TO 155
C 14 A FUNCTION WITH THREE ILL-CONDITIONED MINIMA, A=10**5 (N = 2)
C 14 CONTINUE
A = 1.05
B = 1.0-5
GO TO 155
C 15 A FUNCTION WITH THREE ILL-CONDITIONED MINIMA, A=10**6 (N = 2)
C 15 CONTINUE
A = 1.06
B = 1.0-6
C 15 CONTINUE
XX = X(1)*X(1)
YY = X(2)*X(2)
XY = XX*YY
IF (XX*YY>3.0) PENFUN = XX*YY-H4+R5
RETURN
C 5 CONTINUE
XX = A1*X1
YY = A2*X2
FUNZ = (XX - 0.5*D + PSAX*XX + PSAX*XX + 0.5*D*YY + YY)
RETURN
C 4 A FUNCTION WITH A SINGLE HOM OF LOCAL MINIMA (N = 2)
C 5 CONTINUE
XX = A1*X1
YY = A2*X2
FUNZ = (XX - 0.5*D + PSAX*XX + PSAX*XX + 0.5*D*YY + YY)
RETURN
C 6 SIX-HUMP CAMEL FUNCTION (N = 2)
C CONTINUE
XX = A1*X1
YY = A2*X2
FUNZ = (XX - 0.5*D + PSAX*XX + PSAX*XX + 0.5*D*YY + YY)
RETURN
C 7 TWO-DIMENSIONAL PENALIZED SHUBERT FUNCTION, BETA = 0 (N = 2)
C CONTINUE
XX = A1*X1
YY = A2*X2
GO TO 95
C 8 TWO-DIMENSIONAL PENALIZED SHUBERT FUNCTION, BETA = 0.5 (N = 2)
C CONTINUE
xx = A1*XX
YY = A2*YY
FUNZ = (XX - 0.5*D + PSAX*XX + PSAX*XX + 0.5*D*YY + YY)
RETURN
C 9 CONTINUE
XX = A1*X1
YY = A2*X2
FUNZ = ((XX - 0.5*D + PSAX*XX + PSAX*XX + 0.5*D*YY + YY)
RETURN
C 10 CONTINUE
XX = A1*X1
YY = A2*X2
FUNZ = ((XX - 0.5*D + PSAX*XX + PSAX*XX + 0.5*D*YY + YY)
RETURN
C 11 CONTINUE
XX = A1*X1
YY = A2*X2
FUNZ = ((XX - 0.5*D + PSAX*XX + PSAX*XX + 0.5*D*YY + YY)
RETURN
C 12 CONTINUE
XX = A1*X1
YY = A2*X2
FUNZ = ((XX - 0.5*D + PSAX*XX + PSAX*XX + 0.5*D*YY + YY)
RETURN
C 13 CONTINUE
XX = A1*X1
YY = A2*X2
FUNZ = ((XX - 0.5*D + PSAX*XX + PSAX*XX + 0.5*D*YY + YY)
RETURN
C 14 CONTINUE
XX = A1*X1
YY = A2*X2
FUNZ = ((XX - 0.5*D + PSAX*XX + PSAX*XX + 0.5*D*YY + YY)
RETURN
C 15 CONTINUE
XX = A1*X1
YY = A2*X2
FUNZ = ((XX - 0.5*D + PSAX*XX + PSAX*XX + 0.5*D*YY + YY)
RETURN
C 16 CONTINUE
XX = A1*X1
YY = A2*X2
FUNZ = ((XX - 0.5*D + PSAX*XX + PSAX*XX + 0.5*D*YY + YY)
RETURN
C 17 CONTINUE
XX = A1*X1
YY = A2*X2
FUNZ = ((XX - 0.5*D + PSAX*XX + PSAX*XX + 0.5*D*YY + YY)
RETURN
C 18 CONTINUE
XX = A1*X1
YY = A2*X2
FUNZ = ((XX - 0.5*D + PSAX*XX + PSAX*XX + 0.5*D*YY + YY)
RETURN
C 19 CONTINUE
XX = A1*X1
YY = A2*X2
FUNZ = ((XX - 0.5*D + PSAX*XX + PSAX*XX + 0.5*D*YY + YY)
RETURN
C 20 CONTINUE
XX = A1*X1
YY = A2*X2
FUNZ = ((XX - 0.5*D + PSAX*XX + PSAX*XX + 0.5*D*YY + YY)
RETURN
C }
SUBROUTINE GLOROTF (NPROB, N, X, FUNZ)
C
THE SUBROUTINE GLOROTF PROVIDES THE CODING OF 37 REAL-VALUED
FUNCTIONS OF N REAL VARIABLES, TO BE USED TOGETHER WITH THE
SUBROUTINE GLORIP, TO DEFINE 37 TEST PROBLEMS FOR GLOBAL
MINIMIZATION SOFTWARE.
C
THE SUBROUTINE GLOROTF RETURNS IN FUNZ THE FUNCTION VALUE
AT THE POINT X = (X1, X2, ..., XN) FOR THE FUNCTION DEFINED BY
PROBLEM NUMBER NPROB.
C
CALLING STATEMENT
C
CALL GLOROTF (NPROB, N, X, FUNZ)
C
DESCRIPTION OF THE CALL PARAMETERS
C
THE FORTRAN IMPLICIT TYPE DEFINITION FOR INTGERS IS USED.
ALL NON-INTEGER PARAMETERS ARE DOUBLE-PRECISION.
C
NPROB IS THE (INPUT) TEST-PROBLEM NUMBER.
C
N IS THE (INPUT) DIMENSION OF THE PROBLEM.
C
X IS AN (INPUT) N-VECTOR CONTAINING THE INDEPENDENT VARIABLES.
C
FUNZ IS THE (OUTPUT) VALUE AT X OF THE FUNCTION DEFINED BY
PROBLEM NUMBER NPROB.
C
DOUBLE PRECISION X
DOUBLE PRECISION FUNZ
DOUBLE PRECISION PI
DOUBLE PRECISION PTA
DOUBLE PRECISION P4A
DOUBLE PRECISION P5A
DOUBLE PRECISION P6A
DOUBLE PRECISION P9A, P9B
DOUBLE PRECISION PI7A
DOUBLE PRECISION P20A, P2CB
DOUBLE PRECISION P21A, P21B, P21C, P21D
DOUBLE PRECISION P22A, P22B, P23B, P23C, P23D
DOUBLE PRECISION P3A, P3B, P3C, P3D, P360
DOUBLE PRECISION P7A, P7B, P7C, P7D
DOUBLE PRECISION X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X11, X12
DOUBLE PRECISION X13, X14, X15, X16, X17, X18, X19, X20, X21, X22, X23
DOUBLE PRECISION X24, X25, X26, X27, X28, X29, X30, X31, X32, X33, X34
DOUBLE PRECISION X35, X36, X37, X38, X39, X40, X41, X42, X43, X44, X45
DOUBLE PRECISION X46, X47, X48, X49, X50, X51, X52, X53, X54, X55, X56
DOUBLE PRECISION X57, X58, X59, X60, X61, X62, X63, X64, X65, X66, X67
DOUBLE PRECISION X68, X69, X70, X71, X72, X73, X74, X75, X76, X77, X78
DOUBLE PRECISION X79, X80, X81, X82, X83, X84, X85, X86, X87, X88, X89
DOUBLE PRECISION X90, X91, X92, X93, X94, X95, X96, X97, X98, X99, X100
C
DIMENSION X(N)
C
* VARIOUS CONSTANTS AND NUMBERS*
A global optimization algorithm using stochastic differential equations

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

(submitted to ACM Transactions on Mathematical Software).
A GLOBAL OPTIMIZATION ALGORITHM USING
STOCHASTIC DIFFERENTIAL EQUATIONS

Filippo Aluffi-Pentini
Bari University

Valerio Parisi
2nd Rome University

Francesco Zirilli
Salerno University

The research reported in this paper has been made possible through the support and sponsorship of the U.S. Government through its European Research Office of the U.S. Army under contract n. DAJA-37-81-C-0740 with the University of Camerino, Italy.

Authors' Addresses:

F. Aluffi-Pentini, Dipartimento di Matematica, Università di Bari, 70123 Bari (Italy);
V. Parisi, Istituto di Fisica, 2nd Università di Roma "Tor Vergata", Via Orazio Raimondo, 00173 (La Romanina) Roma (Italy);
F. Zirilli, Istituto di Matematica, Università di Salerno, 84100 Salerno (Italy).
ABSTRACT

SIGMA is a set of FORTRAN subprograms for solving the global optimization problem, which implement a method founded on the numerical solution of a Cauchy problem for stochastic differential equations inspired by quantum physics.

This paper gives a detailed description of the method as implemented in SIGMA, and reports on the numerical tests which have been performed while the SIGMA package is described in the accompanying Algorithm.

The main conclusions are that SIGMA performs very well on several hard test problems; unfortunately given the state of the mathematical software for global optimization it has not been possible to make conclusive comparisons with other packages.

Categories and Subject Descriptors:
G.1.0 [Numerical Analysis]: Optimization - ;
G.4 [Mathematical Software]: Algorithm analysis; certification and Testing.

General terms: Algorithms, Theory, Verification

Additional Key Words and Phrases: Global Optimization, Stochastic Differential Equations.
1. Introduction.

In [1] a method for solving the global optimization problem was proposed. The method associates a stochastic differential equation with the function whose global minimizer we are looking for.

The stochastic differential equation is a stochastic perturbation of a "steepest descent" ordinary differential equation and is inspired by quantum physics. In [1] the problem of the numerical integration of the stochastic equations introduced was considered and a suitable "stochastic" variation of the Euler method was suggested.

SIGMA is a set of FORTRAN subprograms implementing the above method.

In sect. 2 we describe the method as implemented in SIGMA; in sect. 3 we give a general description of the method and some details on the implementation; in sect. 4 some numerical experience on test problems is presented and in sect. 5 conclusions are given.

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

The SIGMA package and its usage are described in the accompanying Algorithm.
2. The method.

Let \( \mathbb{R}^N \) be the \( N \)-dimensional real euclidean space and let \( f: \mathbb{R}^N \rightarrow \mathbb{R} \) 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 \( x^* \in \mathbb{R}^N \) (or possible one of the points) such that

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

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 \( g(x) = 0 \), where \( g: \mathbb{R}^N \rightarrow \mathbb{R}^N \) can be formulated as a global optimization problem 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 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 (Itô) stochastic differential equation

\[
\frac{d\xi}{d\beta} = -\nabla f(\xi) dt + dw
\]

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

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 \( \xi \), where \( \xi = \sqrt{\frac{2kT}{m}} \), \( T \) is the absolute temperature, \( k \) the Boltzmann constant, \( m \) a suitable mass coefficient, and \( f \) is the potential energy.

We assume that

\[
\lim_{\|x\| \to \infty} f(x) = +\infty
\]

in such a way that:

\[
\int_{\mathbb{R}} e^{-\frac{1}{2} f(x)} \, dx < \infty \quad \forall x \in (\mathbb{R}\setminus\{0\})
\]

and that the minimizers of \( f \) are isolated and non degenerate.

It is well known that if \( \xi^c(t) \) is the solution process of (2.2) starting from an initial point \( x_0 \), the probability density function \( p^{\xi^c}(t, x) \) of \( \xi^c(t) \) approaches as \( t \to \infty \) the limit density \( p^c_\infty(x) \) where

\[
p^c_\infty(x) = \Lambda_c \exp \left( -\frac{1}{2} f(x) \right)
\]

where \( \Lambda_c \) is a normalization constant. The way in which \( p^c(t, x) \) for a class of one-dimensional systems approaches \( p^c_\infty(x) \) has been studied in detail by considering the spectrum of the corresponding Fokker-Planck operators in [8].
We note that \( u \) is independent of \( x \), and that as \( \epsilon \to 0 \) \( p_\epsilon \)
becomes more concentrated at the global minimizers of \( f \). That is,

\[
\lim_{\epsilon \to 0} f(t) = \bar{f}, \quad \text{in law}
\]

where \( \bar{f} \) has a probability density given by (2.5) and

\[
\lim_{\epsilon \to 0} f(T) = \bar{f}, \quad \text{in law}
\]

where \( \bar{f} \) 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 \( x = 1 \) and \( f \) has two global minimizers \( x_1, x_2 \), with
\[
\frac{df}{dx}(x_i) = c_i > 0, \quad i = 1, 2,
\]
we have (in distribution sense)

\[
\lim_{\epsilon \to 0} \lim_{\epsilon \to 0} (x-x_i) = \gamma \cdot (x-x_i)
\]

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

That is, we want to consider:

\[
\lim_{\epsilon \to 0} \lim_{t \to \infty} \left( x(t) - x_i \right) = \gamma \cdot (x-x_i)
\]

with initial condition

\[
\lim_{\epsilon \to 0} (x(0)) = x_i
\]

where

\[
\lim_{\epsilon \to 0} \quad \text{for} \quad t \to \infty.
\]
In physical terms condition (2.11) means that the temperature \( T \) is decreased to 0 (absolute zero) when \( t \to \infty \), that is, the system is "frozen".

Since we want to end up in a global minimizer of \( f \), that is, a global minimizer of the (potential) energy, the system has to be frozen very slowly (adiabatically). The way in which \( \zeta(t) \) must go to zero, in order to have that when \( t \to \infty \) the solution \( \zeta(t) \) 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 \( \zeta(t) \) to a random variable concentrated at the global minimizers of \( f \), and the convergence with probability one of the paths of \( \zeta(t) \) 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 \( \zeta(t_k) \) is approximated by the solution \( \zeta_k \) of the following finite difference equations:
\[ (2.12) \quad \frac{z_{k+1} - z_k}{h_k} = -h_k \nabla f(z_k) + \mathcal{O}(h_k)(w_{k+1} - w_k) \quad k = 0, 1, 2, \ldots \]

\[ (2.13) \quad z_0 = x_0 \]

where \( t_i = 0, \quad t_k = \sum_{i=1}^{k-1} h_i, \quad h_k = 0, \quad \) and \( w_k = w(t_k) = k = 0, 1, 2, \ldots \).

The computationally cheap Euler step seems a good choice here since in order to obtain the global minimizers of \( f \) as asymptotic values of the paths \( z(t) \) should go to zero very slowly when \( t \to \infty \), and therefore a large number of time integration steps must be computed.

On the right hand side of (2.12) we add the random term \( \mathcal{O}(h_k)(w_{k+1} - w_k) \) to the deterministic term \( -h_k \nabla f(z_k) \), 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 \( \nabla f(z_k) \) is frequently lost.

In order to avoid this inconvenience we substitute the gradient \( \nabla f(z_k) \) with a "random gradient" as follows. Let \( \mathbf{r} \) be an \( N \)-dimensional random vector of length 1 uniformly distributed on the \( N \)-dimensional unit sphere. Then for any given (non-random) vector \( \mathbf{v} \in \mathbb{R}^N \) its projection along \( \mathbf{r} \) is such that:

\[ (2.14) \quad N \cdot \mathbf{r} = \mathbb{E}(\langle \mathbf{r}, \mathbf{v} \rangle \mathbf{r}) = \mathbf{v} \]

where \( \mathbb{E}(\cdot) \) 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 \( \nabla f(z_k) \) with the "random gradient"
(2.15) \[ \frac{d}{-k} \frac{1}{N} \neq r, \text{if} \frac{1}{-k} > r. \]

We note that since \( \frac{1}{N} \cdot \frac{d}{-k} \) is the directional derivative in the direction \( r \), it is computationally much cheaper (e.g. when forward differences are used, only 2 function evaluations are needed to approximate \( \frac{1}{N} \cdot \frac{d}{-k} \)). Therefore, the paths are computed approximating \( \frac{d}{-k}(t_k) \) with the solution \( x_k \) of the following differences equations:

(2.16) \[ \frac{1}{-k+1} \frac{1}{-k} = -h_k \cdot \frac{1}{-k} \cdot (t_k) (w_{k+1} - w_k) \quad k = 0, 1, 2, \ldots \]

(2.17) \[ x_0 = x_0 \]

where \( \frac{d}{-k}(t_k) \) is a finite difference (forward or central) approximation to \( \frac{d}{-k}(t_k) \).

The complete algorithm is described in the next section.
7. The complete algorithm.

We give in sect. 5.1 a general description of the algorithm, while implementation details are given in sect. 5.2.

5.1. 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 $N_{\text{TRAJ}}$ of trajectories, which start at time zero from the same initial conditions with possibly different values of $\psi_0$ (note that even if the starting values $x(0)$ are equal the trajectories evolve differently due to the stochastic nature of the integration steps).

The trajectories evolve (simultaneously but independently) during an "observation period" having a given duration (sect. 3.2.5), and within which the noise coefficient of each trajectory is kept at a constant value $\eta$, while the values of the steplength $h_k$ and of the spatial discretization increment $\Delta x_k$ for computing the random gradient (eq. 2.1[7] and sect. 3.2.2) are automatically adjusted for each trajectory by the algorithm (sects. 5.2.3 and 5.2.4).

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. 5.2.6), that is for generating also a second continuation trajectory differing from the first one only in the starting values for $\psi$ and $x_k$ (sect. 5.2.7), 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.

5.2 Implementation details.

5.2.1 The time-integration step.

The basic time-integration step (eq. (2.16)) is used, for the trajectory under consideration, in the form

$$\tilde{x}_{k+1} = \tilde{x}_k - h_k \lambda_k + \varepsilon_p \sigma h_k \tilde{u}_k (k = 0,1,2, \ldots)$$

where $h_k$ and $\varepsilon_p$ are the current values of the step length and of the noise coefficient (the noise coefficient has a constant value $\varepsilon_p$ throughout the current observation period (sect. 3.1)); $\tilde{u}_k$ is a random vector sample from an N-dimensional standard Gaussian distribution, and

$$\frac{\sigma}{h_k} \tilde{u}_k = \tilde{u}_{k+1} - \tilde{u}_k$$

due to the properties of the Wiener process.

The computation of the finite-differences random gradient $\frac{\partial f}{\partial x}$ is described in the next section.
The basic step (3.2.1.1) is actually performed in two half-steps

\[ i_k = i_k - h_k \tilde{z}_k \]  \hspace{1cm} \text{(first half-step)}

and

\[ i_{k+1} = i_k + h_k \tilde{u}_k \]  \hspace{1cm} \text{(second half-step)}

Both half-steps depend on \( h_k \) while the first depends also on the current value \( x_k \) of the spatial discretization increment used in computing \( \tilde{z}_k \).

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 \( x_k \) of the spatial discretization increment for the trajectory under consideration, we consider the random increment vector

\[ \Delta x_k = x_k + r_k \]

where \( r_k \) is a random sample of a vector uniformly distributed on the unit sphere in \( \mathbb{R}^n \), the forward and central differences

\[
\begin{align*}
\delta_+ f_k &= f(x_k + \Delta x_k) - f(x_k) \\
\delta_- f_k &= f(x_k - \Delta x_k) - f(x_k)
\end{align*}
\]

the forward- and central-differences directional derivatives

\[
\begin{align*}
\delta_+ f_k &= \frac{f(x_k + \Delta x_k) - f(x_k)}{\Delta x_k} \\
\delta_- f_k &= \frac{f(x_k - \Delta x_k) - f(x_k)}{\Delta x_k}
\end{align*}
\]

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. 4.1.

4.1. Test problems.

The set of test problems is fully described in [19] 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 \(\varepsilon = 0\) \((N = 2)\)
8. Two-dimensional penalized Shubert function \(\varepsilon = 0.5\) \((N = 2)\)
9. Two-dimensional penalized Shubert function \(\varepsilon = 1\) \((N = 2)\)
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 \(a = 10^4\) \((N = 2)\)
15. A function with three ill-conditioned minima \(a = 10^5\) \((N = 2)\)
16. Goldstein-Price function \((N = 2)\)
17. Penalized Branin function \((N = 2)\)
18. Penalized Shekel function \(M = 5\) \((N = 1)\)
a) Relative difference criterion

\[ |x - y| \leq \tau_{\text{REL}} \left( |x| + |y| \right)/2 \]

b) Absolute difference criterion

\[ |x - y| \leq \tau_{\text{ABS}} \]

where \( \tau_{\text{REL}} \) and \( \tau_{\text{ABS}} \) are given non-negative tolerances.
Let $\lambda_1$ be the largest eigenvalue of the (symmetric and non-negative definite) matrix $C$.

We adopt the updating matrix 

$$F_A = B\lambda_1 I - C$$

where $I$ is the $N \times N$ identity matrix, $B > 1$ ($B = 1.3$ in the present implementation), and we obtain the updated value $A'$ of $A$ by means of the formula 

$$A' = \alpha A F_A$$

where $\alpha$ 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 $B > 1$, it has the same set of eigenvectors as $C$, its eigenvalue spectrum is obtained from the spectrum of $C$ by reflection around $\lambda = \frac{B\lambda_1}{2}$, and it therefore acts in the right direction to counter the ill-conditioning of $f$.

The magnitude of the counter-effect depends on $B$: the adopted value has been experimentally adjusted.

The updated bias vector $b'$ is chosen in order that the scaling at $x$ does not alter $\bar{x}$, i.e. in order that 

$$A'x + b' = Ax + b.$$ 

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 \( \tilde{x} = Ax + b \), where \( A \) is the rescaling matrix and \( b \) is a bias vector, and, instead of \( f(x) \), we minimize with respect to \( \tilde{x} \) the function \( f(x) = f(Ax + b) \), and we try to counter the ill-conditioning of \( f \) with respect to \( \tilde{x} \) by suitably adjusting \( A \) (and \( b \) is adjusted in order not to alter \( x \)).

The updating of \( A \) is obtained by means of an updating matrix \( F_A \) 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 \( K_{\text{pass}} \) and greater than \( 2N \).

The updating matrix \( F_A \) is computed as described below, keeping in mind that the random gradients are the only simply usable data on the behavior of \( f \) computed by the algorithm.

Let \( \tilde{r}(i), \ i = 1, 2, \ldots, N \), be the column vectors of the components of all the \( N \) finite-difference random gradients \( \tilde{r}^F \) or \( \tilde{r}^G \) evaluated along the trajectory (also for rejected steps) from the last scaling.

If sufficient data are available (i.e., if \( N \geq 2N^2 \)) we compute the average

\[
\tilde{r} = \frac{1}{N} \sum_{i=1}^{N} \tilde{r}(i)
\]

and the estimated covariance matrix

\[
\tilde{C} = \frac{1}{N} \sum_{i=1}^{N} [\tilde{r}(i) - \tilde{r}] [\tilde{r}(i)^T - \tilde{r}^T]
\]

which seems to be a reasonable indicator, given the available data, of the average ill-conditioning of \( f \), as having the larger eigenvalues associated with the directions along which the second directional derivatives \( \tilde{r}^T \tilde{C}^{-1} \tilde{r} \), 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

\[
\mathcal{R}_i^{\text{MIN}} \leq x_i \leq \mathcal{R}_i^{\text{MAX}}, \quad i = 1, 2, \ldots, N,
\]

where the interval boundaries must be given before trial start.

Outside the admissible region the function \( f(x) \) is replaced by an exponentially increasing function, in such a way that the values of \( f \) and of the external function are matched at the boundary of the region.

3.2.12 Scaling.

In order to make ill-conditioned problems more tractable, rescaling is performed by the algorithm as follows.
the preceding trial, according to the outcome (stopping condition) of the preceding trial and to the number $t$ of trials performed from algorithm start, as compared to the given maximum number of trials $N_{\text{TRIAL}}$

successful stop: $t = 10^3$

unsuccessful uniform stop:

$t = 10^3$ if $t = \lfloor (2/3) N_{\text{TRIAL}} \rfloor$

$t = 10^2$ otherwise,

where $\lfloor x \rfloor$ is the smallest integer not smaller than $x$

unsuccessful non-uniform stop: $t = 10^2$

The initial point $x_2$ is selected as follows:

if $t = \lfloor (2/3) N_{\text{TRIAL}} \rfloor$ take the value of $x_2$ at algorithm start

otherwise take $x_2 = x_{\text{opt}}$

where $x_{\text{opt}}$ 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 $dw$ which are the values reached at the end of the preceding trial.

5.2.10 Stopping criteria for the algorithm.

The complete algorithm is stopped, at the end of a trial, if a given number $N_{\text{SUC}}$ has been reached of uniform trial stops all at the current $\text{opt}$ level, or in any case if a maximum given number $N_{\text{TRIAL}}$ of trials has been reached.

Success is claimed by the algorithm if at least one uniform stop occurred at the current $\text{opt}$ level.
and the best minimum function value $f_{\text{OPT}}$ found so far from algorithm start: if $f_{\text{TMIN}}$ and $f_{\text{OPT}}$ satisfy at least one of the above criteria, with the same tolerances, the trial is considered successful at the level $f_{\text{OPT}}$; otherwise the trial is again considered unsuccessful.

Checking of the stopping criteria is activated only if a minimum given number $N_{\text{MIN}}$ 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 $\varepsilon_p$ for the second continuation of a branched trajectory
- value of $\varepsilon_p$ at trial start
- initial point $x_i$.

The maximum duration of a trial, i.e. the maximum number $N_{\text{MAX}}$ 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 $N_{\text{MAX}}$

The policy for selecting $\varepsilon_p$ for the second continuation of a branched trajectory was described in sect. 3.2.7.

The value of $\varepsilon_p$ at the start of a new trial is obtained by means of a multiplicative updating factor $\alpha$ applied to the starting value of
The updating factor $E_p$ for $p$ is as follows:

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

$$E = 10^{x^2} \text{ where } x \text{ is a random sample from a standard normal distribution}$$

for all other trials

$$E = 2^{y^2} \text{ where } y \text{ is a random sample from a standard Cauchy distribution, i.e. with density}$$

$$f(y) = 1/(1+y^2)$$

The updating factor for $X_k$ is:

$$E_{X_k} = 10^{z^2} \text{ where } z \text{ is a random sample from a standard normal distribution.}$$

3.2.8 Stopping criteria for a trial.

A trial is stopped, at the end of an observation period, and after having discarded the worst trajectory, if all the final function values of the remaining trajectories (possibly at different points $x$) are "numerically equal", i.e. if the maximum, $f_{\text{MAX}}$, and the minimum, $f_{\text{MIN}}$, among the trial final values satisfy at least one of the criteria in sect. 3.2.13, the relative difference criterion with a given stopping tolerance $\text{TREL}$ and/or the absolute difference criterion with given stopping tolerance $\text{ABS}$ ("uniform stop at the level $f_{\text{MIN}}$."

The trial is also anyway stopped, at the end of the observation period, if a maximum given number $X_{\text{MAX}}$ 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 $f_{\text{MIN}}$
From the point of view of the noise coefficient \( p \), a trajectory with larger \( p \) is considered better if the comparison is made in an early observation period \( k_p \) as long as \( k_p \geq M_p \cdot l_b \), where \( k_p \) is the number of elapsed observation periods, and \( M_p, l_b \) 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 \( l_b \) (a given integer); exceptional branching, where the best trajectory is selected, occurs for the first time at the end of observation period \( k_{po} \), and then every \( M_p \) periods (\( k_{po} \) and \( M_p \) are given integers); i.e. exceptional observation periods are those numbered

\[
k_p = k_{po} + M_p \cdot i = 1, 2, \ldots \]

5.2.7 The second continuation of a branched trajectory.

While the first (unperturbed) continuation of a trajectory that undergoes branching starts with the current values of \( p \) and \( x_k \), the second continuation starts with values obtained by means of multiplicative random updating factors applied to the current values.
In phase 6a: \( t = 0.1 \)

We finally remark that \( h_k \) and \( x_k \) are bounded by suitable constants to avoid computational failures.

5.2.3 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) \( N_{hp} = 1 + \lfloor \log_2(k_p) \rfloor \) ("short" duration)
2) \( N_{hp} = k_p^2 \) ("medium-size" duration)
3) \( N_{hp} = k_p \) ("long" duration)

where \( k_p = 1, 2, \ldots \), and \( \lfloor x \rfloor \) 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 $f_k = f_k + \Delta f_k$ for numerical equality according to the relative difference criterion (sect. 3.2.13) with tolerances $R_1 = 10^{-12}$ and $R_2 = 10^{-9}$, and take

$\varepsilon = 2$ if $f_k$ and $f_k$ are "equal" within $R_1$

$\varepsilon = 1$ if $f_k$ and $f_k$ are not "equal" within $R_2$

$\varepsilon = 1$ otherwise.

The interval $(10^{-12}, 10^{-9})$ 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 $\varepsilon$ for $h_k$

In phase 4a:

$\varepsilon = 1/1.05$ for the first attempt to the first half-step

$\varepsilon = 1/2$ for the second attempt

$\varepsilon = 1/10$ for all other attempts

In phase 5 the value of $\varepsilon$ 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 $r > 0$

$\varepsilon = 1$ (if $a \leq 2r$)

$\varepsilon = 1.1$ (if $2r < a \leq 5r$)

$\varepsilon = 2$ (if $5r < a$)

If $r = 0$

$\varepsilon = 2$ (if $a = 1$)

$\varepsilon = 10$ (if $a > 1$)
If the half-step is rejected: reject also the first half-step, update (decrease) $h_k$, and go back to 1.

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 $10^{-10}$)).

5.2.4. The updating of $h_k$ and $x_k$.

The time-integration steplength $h_k$ and the spatial discretization increment $x_k$ 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 for $x_k$:
  - In phase 1b: $\epsilon = 10^1$.
  - In phase 2a: $\epsilon = 10$.
  - In phase 1b: $\epsilon = 10^{17}$.

In phase 3 the value of $\epsilon$ depends on the magnitude of the current estimated function increment $f_k = f_k(x_k)$ where $x_k$ is $x_k^{1}$ or $x_k^{2}$ as appropriate, and the function value $f_k = f_k(x_k)$. 


All attempts are with the current (i.e. updated) values of $h_k$ and $\Delta x_k$.

The sequence of attempts is as follows:

1. Pick up a random unit vector $\frac{r_k}{\|r_k\|}$.
   1a. Compute the random increment $\frac{s_k}{\|s_k\|}$ (sect. 3.2.2).
   1b. If $\frac{s_k}{\|s_k\|}$ (and therefore $\frac{\Delta x_k}{\|\Delta x_k\|}$) is too small (i.e. if the square of the Euclidean norm of the difference between the computed values of $\frac{x_k + s_k}{\|x_k + s_k\|}$ and $\frac{x_k}{\|x_k\|}$ is zero, due to the finite arithmetics of the machine): update (increase) $\Delta x_k$ and go back to 1a.

2. Compute $\frac{F_k}{\|F_k\|}$ (eq. (3.2.2.2)).
   2a. If the computed value of $(\frac{F_k}{\|F_k\|})^2$ is zero (due to the finite arithmetics): update (increase) $\Delta x_k$ and go back to 1a.

3. Compute the first half-step with $\frac{F_k}{\|F_k\|}$.
   Compute $\frac{F_k}{\|F_k\|}$ (eq. (3.2.5.1)).
   3a. If $\frac{F_k}{\|F_k\|} \leq \frac{F_k}{\|F_k\|} \Delta x_k$
       accept the first half-step and jump to 5.

4. Compute the first half-step with $\frac{C_k}{\|C_k\|}$ to check the appropriateness of $\Delta x_k$.
   Compute $\frac{C_k}{\|C_k\|}$ (eq. (3.2.5.1)).
   4a. If $\frac{C_k}{\|C_k\|} \leq \frac{C_k}{\|C_k\|} \Delta x_k$
       reject the half-step, update (decrease) $h_k$, and go back to 1.
   4b. Otherwise: accept the half-step, and update (decrease) $\Delta x_k$.

5. Update (increase) $h_k$.

6. Compute the second half-step.
   Compute $\frac{C_k}{\|C_k\|}$ (eq. (3.2.5.2)).
and the forward- and central-differences random gradients

\[(3.2.2.3) \quad \frac{\Delta F}{\Delta k} = N \frac{\Delta r}{\Delta k}, \quad \frac{\Delta C}{\Delta k} = N \frac{\Delta r}{\Delta k}\]

We use \(\frac{\Delta F}{\Delta k}\) or \(\frac{\Delta C}{\Delta k}\) for \(\Delta f_k\) 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 \(\frac{\Delta F}{\Delta k}\) or \(\frac{\Delta C}{\Delta k}\) eq. (3.2.2.3)) as described below.

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

\[(3.2.3.1) \quad \Delta f_k = f(\tau_k') - f(\tau_k)\]

Since \(\Delta f_k\) should be non-positive for a sufficiently small value of \(\Delta h_k\), 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

\[(3.2.3.2) \quad \Delta f_k = f(\tau_{k+1}) - f(\tau_k)\]

is positive and too large (greater than 100 \(\frac{\tau}{p}\) in the present implementation).

The sequence of attempts affects the updating of \(h_k\) and \(x_k\) as described below; the amount of the updating is described in sect. 5.2.4.

12

and the forward- and central-differences random gradients

\[(3.2.2.3) \quad \frac{\Delta F}{\Delta k} = N \frac{\Delta r}{\Delta k}, \quad \frac{\Delta C}{\Delta k} = N \frac{\Delta r}{\Delta k}\]

We use \(\frac{\Delta F}{\Delta k}\) or \(\frac{\Delta C}{\Delta k}\) for \(\Delta f_k\) 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 \(\frac{\Delta F}{\Delta k}\) or \(\frac{\Delta C}{\Delta k}\) eq. (3.2.2.3)) as described below.

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

\[(3.2.3.1) \quad \Delta f_k = f(\tau_k') - f(\tau_k)\]

Since \(\Delta f_k\) should be non-positive for a sufficiently small value of \(\Delta h_k\), 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

\[(3.2.3.2) \quad \Delta f_k = f(\tau_{k+1}) - f(\tau_k)\]

is positive and too large (greater than 100 \(\frac{\tau}{p}\) in the present implementation).

The sequence of attempts affects the updating of \(h_k\) and \(x_k\) as described below; the amount of the updating is described in sect. 5.2.4.
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 GLMTF and GLMIP, which are available in [40].
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 $10^{-38}$ and $10^{+38}$. The tests were actually performed on:

- UNIVAC 1100/82 with EXEC8 operating system and FORTRAN (ASCII) computer (level 1OR1) ("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 $N_{SU} = 1, 2, 3, 4, 5$. All numerical values used for the parameters are set in the driver SIGMA1 and in the other subroutines which are described in the accompanying Algorithm.

All numerical results are reported on Tables 1, 2, and 3. Table 1 reports some performance data (i.e. output indicator IOUT and number of functions evaluations) as obtained from SIGMA output for each of the 37 test problems and for the testing both on the "large" and "small" dynamic range machines. In order to evaluate the performance of SIGMA we consider all the cases in which the program claimed a success (output indicator IOUT > 0) or a failure (IOUT ≤ 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

<table>
<thead>
<tr>
<th>N_SUC</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>N</td>
<td>Nf</td>
<td>Ie</td>
<td>Nf</td>
<td>Ie</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>5,588</td>
<td>0</td>
<td>11,467</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>3,254</td>
<td>0</td>
<td>9,509</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>8,058</td>
<td>0</td>
<td>17,741</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>6,594</td>
<td>0</td>
<td>15,898</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>12,080</td>
<td>0</td>
<td>23,221</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>2,067</td>
<td>0</td>
<td>8,343</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>2</td>
<td>32,185</td>
<td>0</td>
<td>35,256</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>5,600</td>
<td>0</td>
<td>347,039</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>2</td>
<td>6,180</td>
<td>0</td>
<td>83,625</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>3,596</td>
<td>0</td>
<td>6,731</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>2</td>
<td>3,191</td>
<td>0</td>
<td>8,384</td>
<td>0</td>
</tr>
<tr>
<td>12</td>
<td>2</td>
<td>4,799</td>
<td>0</td>
<td>7,296</td>
<td>0</td>
</tr>
<tr>
<td>13</td>
<td>2</td>
<td>7,105</td>
<td>0</td>
<td>10,287</td>
<td>0</td>
</tr>
<tr>
<td>14</td>
<td>2</td>
<td>6,671</td>
<td>0</td>
<td>10,654</td>
<td>0</td>
</tr>
<tr>
<td>15</td>
<td>2</td>
<td>7,747</td>
<td>0</td>
<td>11,631</td>
<td>0</td>
</tr>
<tr>
<td>16</td>
<td>2</td>
<td>16,021</td>
<td>0</td>
<td>26,560</td>
<td>0</td>
</tr>
<tr>
<td>17</td>
<td>2</td>
<td>2,700</td>
<td>0</td>
<td>6,670</td>
<td>0</td>
</tr>
<tr>
<td>18</td>
<td>4</td>
<td>4,674</td>
<td>0</td>
<td>16,556</td>
<td>0</td>
</tr>
<tr>
<td>19</td>
<td>4</td>
<td>4,759</td>
<td>0</td>
<td>54,559</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>4</td>
<td>9,955</td>
<td>0</td>
<td>1,092</td>
<td>0</td>
</tr>
<tr>
<td>21</td>
<td>3</td>
<td>3,416</td>
<td>0</td>
<td>5,199</td>
<td>0</td>
</tr>
<tr>
<td>22</td>
<td>2</td>
<td>4,725</td>
<td>0</td>
<td>11,88</td>
<td>0</td>
</tr>
<tr>
<td>23</td>
<td>2</td>
<td>12,880</td>
<td>0</td>
<td>13,000</td>
<td>0</td>
</tr>
<tr>
<td>NSPEC</td>
<td>N</td>
<td>Nf</td>
<td>Ie</td>
<td>Nf</td>
<td>Ie</td>
</tr>
<tr>
<td>-------</td>
<td>---</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
</tr>
<tr>
<td></td>
<td>24</td>
<td>3</td>
<td>8,099</td>
<td>0</td>
<td>36,057</td>
</tr>
<tr>
<td></td>
<td>25</td>
<td>4</td>
<td>11,954</td>
<td>0</td>
<td>54,212</td>
</tr>
<tr>
<td></td>
<td>26</td>
<td>5</td>
<td>43,083</td>
<td>0</td>
<td>281,104</td>
</tr>
<tr>
<td></td>
<td>27</td>
<td>8</td>
<td>2,324</td>
<td>0</td>
<td>21,124</td>
</tr>
<tr>
<td></td>
<td>28</td>
<td>10</td>
<td>50,975</td>
<td>0</td>
<td>426,171</td>
</tr>
<tr>
<td></td>
<td>29</td>
<td>2</td>
<td>25,462</td>
<td>0</td>
<td>35,675</td>
</tr>
<tr>
<td></td>
<td>30</td>
<td>3</td>
<td>15,734</td>
<td>0</td>
<td>113,789</td>
</tr>
<tr>
<td></td>
<td>31</td>
<td>4</td>
<td>11,516</td>
<td>0</td>
<td>143,757</td>
</tr>
<tr>
<td></td>
<td>32</td>
<td>5</td>
<td>50,911</td>
<td>0</td>
<td>176,840</td>
</tr>
<tr>
<td></td>
<td>33</td>
<td>6</td>
<td>53,178</td>
<td>0</td>
<td>102,652</td>
</tr>
<tr>
<td></td>
<td>34</td>
<td>7</td>
<td>14,594</td>
<td>0</td>
<td>298,256</td>
</tr>
<tr>
<td></td>
<td>35</td>
<td>5</td>
<td>33,635</td>
<td>0</td>
<td>50,348</td>
</tr>
<tr>
<td></td>
<td>36</td>
<td>2</td>
<td>3,102</td>
<td>0</td>
<td>10,176</td>
</tr>
<tr>
<td></td>
<td>37</td>
<td>5</td>
<td>6,938</td>
<td>0</td>
<td>12,469</td>
</tr>
<tr>
<td>N_{PUBS}</td>
<td>N</td>
<td>Xf</td>
<td>Ie</td>
<td>Nf</td>
<td>Ie</td>
</tr>
<tr>
<td>---------</td>
<td>---</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>7,522</td>
<td>0</td>
<td>12,657</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>3,131</td>
<td>0</td>
<td>6,542</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>11,526</td>
<td>0</td>
<td>15,342</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>9,265</td>
<td>0</td>
<td>17,713</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>12,094</td>
<td>0</td>
<td>19,716</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>4,650</td>
<td>0</td>
<td>11,040</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>2</td>
<td>10,543</td>
<td>0</td>
<td>44,408</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>27,044</td>
<td>0</td>
<td>76,348</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>2</td>
<td>24,348</td>
<td>0</td>
<td>35,885</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>4,114</td>
<td>0</td>
<td>9,959</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>2</td>
<td>3,254</td>
<td>0</td>
<td>7,901</td>
<td>0</td>
</tr>
<tr>
<td>12</td>
<td>2</td>
<td>6,711</td>
<td>0</td>
<td>9,949</td>
<td>0</td>
</tr>
<tr>
<td>13</td>
<td>2</td>
<td>6,771</td>
<td>0</td>
<td>11,031</td>
<td>0</td>
</tr>
<tr>
<td>14</td>
<td>2</td>
<td>6,208</td>
<td>0</td>
<td>9,443</td>
<td>0</td>
</tr>
<tr>
<td>15</td>
<td>2</td>
<td>6,313</td>
<td>0</td>
<td>13,581</td>
<td>0</td>
</tr>
<tr>
<td>16</td>
<td>2</td>
<td>5,439</td>
<td>0</td>
<td>10,491</td>
<td>0</td>
</tr>
<tr>
<td>17</td>
<td>2</td>
<td>2,790</td>
<td>0</td>
<td>11,006</td>
<td>0</td>
</tr>
<tr>
<td>18</td>
<td>4</td>
<td>2,446</td>
<td>0</td>
<td>36,252</td>
<td>0</td>
</tr>
<tr>
<td>19</td>
<td>4</td>
<td>4,778</td>
<td>0</td>
<td>19,951</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>4</td>
<td>4,741</td>
<td>0</td>
<td>11,312</td>
<td>0</td>
</tr>
<tr>
<td>21</td>
<td>3</td>
<td>4,334</td>
<td>0</td>
<td>27,816</td>
<td>0</td>
</tr>
<tr>
<td>22</td>
<td>6</td>
<td>3,975</td>
<td>0</td>
<td>8,613</td>
<td>0</td>
</tr>
<tr>
<td>23</td>
<td>5,557</td>
<td>0</td>
<td>8,613</td>
<td>0</td>
<td>24,469</td>
</tr>
</tbody>
</table>

Table 1 (continued)
Table 1 (continued)

<table>
<thead>
<tr>
<th>NPROB</th>
<th>N</th>
<th>Nf</th>
<th>Ie</th>
<th>Nf</th>
<th>Ie</th>
<th>Nf</th>
<th>Ie</th>
<th>Nf</th>
<th>Ie</th>
<th>Nf</th>
<th>Ie</th>
</tr>
</thead>
<tbody>
<tr>
<td>24</td>
<td>5</td>
<td>10,010</td>
<td>0</td>
<td>18,170</td>
<td>0</td>
<td>80,567</td>
<td>0</td>
<td>109,726</td>
<td>0</td>
<td>190,920</td>
<td>0</td>
</tr>
<tr>
<td>25</td>
<td>4</td>
<td>10,657</td>
<td>0</td>
<td>37,655</td>
<td>0</td>
<td>50,285</td>
<td>0</td>
<td>129,598</td>
<td>0</td>
<td>227,082</td>
<td>0</td>
</tr>
<tr>
<td>26</td>
<td>5</td>
<td>59,689</td>
<td>0</td>
<td>369,912</td>
<td>0</td>
<td>460,779</td>
<td>0</td>
<td>470,325</td>
<td>0</td>
<td>585,249</td>
<td>0</td>
</tr>
<tr>
<td>27</td>
<td>8</td>
<td>198,955</td>
<td>0</td>
<td>259,950</td>
<td>0</td>
<td>302,092</td>
<td>0</td>
<td>310,718</td>
<td>0</td>
<td>330,192</td>
<td>0</td>
</tr>
<tr>
<td>28</td>
<td>10</td>
<td>43,466</td>
<td>0</td>
<td>303,550</td>
<td>0</td>
<td>411,854</td>
<td>0</td>
<td>454,095</td>
<td>0</td>
<td>474,422</td>
<td>0</td>
</tr>
<tr>
<td>29</td>
<td>2</td>
<td>15,223</td>
<td>0</td>
<td>55,718</td>
<td>0</td>
<td>95,254</td>
<td>0</td>
<td>131,418</td>
<td>0</td>
<td>171,995</td>
<td>0</td>
</tr>
<tr>
<td>30</td>
<td>3</td>
<td>12,641</td>
<td>0</td>
<td>54,822</td>
<td>0</td>
<td>118,927</td>
<td>0</td>
<td>201,965</td>
<td>0</td>
<td>533,175</td>
<td>0</td>
</tr>
<tr>
<td>31</td>
<td>4</td>
<td>26,235</td>
<td>0</td>
<td>123,716</td>
<td>0</td>
<td>148,183</td>
<td>0</td>
<td>242,535</td>
<td>0</td>
<td>397,066</td>
<td>0</td>
</tr>
<tr>
<td>32</td>
<td>5</td>
<td>35,365</td>
<td>0</td>
<td>86,758</td>
<td>0</td>
<td>186,860</td>
<td>0</td>
<td>285,371</td>
<td>0</td>
<td>316,702</td>
<td>0</td>
</tr>
<tr>
<td>33</td>
<td>6</td>
<td>49,087</td>
<td>0</td>
<td>71,418</td>
<td>0</td>
<td>159,987</td>
<td>0</td>
<td>184,087</td>
<td>0</td>
<td>296,770</td>
<td>0</td>
</tr>
<tr>
<td>34</td>
<td>7</td>
<td>50,237</td>
<td>0</td>
<td>132,768</td>
<td>0</td>
<td>173,955</td>
<td>0</td>
<td>204,081</td>
<td>0</td>
<td>348,809</td>
<td>0</td>
</tr>
<tr>
<td>35</td>
<td>5</td>
<td>14,815</td>
<td>0</td>
<td>53,394</td>
<td>0</td>
<td>79,235</td>
<td>0</td>
<td>129,480</td>
<td>0</td>
<td>206,980</td>
<td>0</td>
</tr>
<tr>
<td>36</td>
<td>2</td>
<td>3,744</td>
<td>0</td>
<td>9,574</td>
<td>0</td>
<td>23,355</td>
<td>0</td>
<td>55,947</td>
<td>0</td>
<td>82,518</td>
<td>0</td>
</tr>
<tr>
<td>37</td>
<td>5</td>
<td>3,847</td>
<td>0</td>
<td>12,239</td>
<td>0</td>
<td>29,411</td>
<td>0</td>
<td>70,599</td>
<td>0</td>
<td>107,264</td>
<td>0</td>
</tr>
</tbody>
</table>

NPROB = problem number given in sect. 4.1.
Ie = 0  success (IOUT > 0) (claimed by SIGMA)
Ie = 1  failure (IOUT < 0) (claimed by SIGMA)
Nf = total number of function evaluations including
the ones needed to compute the "random" gradient
NSUC = see sect. 3.2.10.
<table>
<thead>
<tr>
<th>N_PROB</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>11</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>12</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>13</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>14</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>15</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>16</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>17</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>18</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>19</td>
<td>4</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>20</td>
<td>4</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>21</td>
<td>5</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>22</td>
<td>6</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>23</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>24</td>
<td>5</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>25</td>
<td>4</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>26</td>
<td>5</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>27</td>
<td>8</td>
<td>5</td>
<td>3</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>28</td>
<td>10</td>
<td>4</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
### Table 2 (continued)

**UNIVAC (continued)**

<table>
<thead>
<tr>
<th>NPROB</th>
<th>N</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>29</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>30</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>31</td>
<td>4</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>32</td>
<td>5</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>33</td>
<td>6</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>34</td>
<td>7</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>35</td>
<td>5</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>36</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>37</td>
<td>5</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>
Table 2 (continued)

VAX

<table>
<thead>
<tr>
<th>N_{SUC} = 1 2 3 4 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>NPROB   N       N</td>
</tr>
<tr>
<td>1        1       1</td>
</tr>
<tr>
<td>2        1       1</td>
</tr>
<tr>
<td>3        1       1</td>
</tr>
<tr>
<td>4        2       1</td>
</tr>
<tr>
<td>5        2       1</td>
</tr>
<tr>
<td>6        2       1</td>
</tr>
<tr>
<td>7        2       1</td>
</tr>
<tr>
<td>8        2       3</td>
</tr>
<tr>
<td>9        2       1</td>
</tr>
<tr>
<td>10       2       1</td>
</tr>
<tr>
<td>11       2       1</td>
</tr>
<tr>
<td>12       2       1</td>
</tr>
<tr>
<td>13       2       1</td>
</tr>
<tr>
<td>14       2       1</td>
</tr>
<tr>
<td>15       2       1</td>
</tr>
<tr>
<td>16       2       1</td>
</tr>
<tr>
<td>17       2       1</td>
</tr>
<tr>
<td>18       4       3</td>
</tr>
<tr>
<td>19       4       1</td>
</tr>
<tr>
<td>20       4       1</td>
</tr>
<tr>
<td>21       5       3</td>
</tr>
<tr>
<td>22       6       1</td>
</tr>
<tr>
<td>23       2       1</td>
</tr>
<tr>
<td>24       5       1</td>
</tr>
<tr>
<td>25       4       1</td>
</tr>
<tr>
<td>26       5       1</td>
</tr>
<tr>
<td>27       8       1</td>
</tr>
<tr>
<td>28       10      1</td>
</tr>
</tbody>
</table>
Table 2 (continued)

VAX (continued)

<table>
<thead>
<tr>
<th>NOPROB</th>
<th>N</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>29</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>30</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>31</td>
<td>4</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>32</td>
<td>5</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>33</td>
<td>6</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>34</td>
<td>7</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>35</td>
<td>5</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>36</td>
<td>5</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>37</td>
<td>5</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>

1 = success correctly claimed
2 = failure correctly claimed
3 = incorrect claim
4 = overflow
### TABLE 3

| $N_{SU}C$ | 1  | 2  | 3  | 4  | 5  | | 1  | 2  | 3  | 4  | 5  |
|-----------|----|----|----|----|----| |----|----|----|----|----|
| Totals 1  | 26 | 32 | 34 | 35 | 35 | | 32 | 34 | 34 | 34 | 35 |
| 2         | 0  | 0  | 0  | 0  | 0  | | 0  | 0  | 0  | 0  | 0  |
| 3         | 11 | 5  | 3  | 2  | 2  | | 5  | 3  | 3  | 3  | 2  |
| 4         | 0  | 0  | 0  | 0  | 0  | | 0  | 0  | 0  | 0  | 0  |

1 = success correctly claimed  
2 = failure correctly claimed  
3 = incorrect claim  
4 = overflow
5. Conclusions.

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

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

Table 2 shows that from the point of view of the effectiveness measured by the number of correctly claimed successes the performance of SIGMA is very satisfactory; moreover, it is remarkably machine independent (note that completely different pseudo-random numbers sequences are generated by the algorithm on the two test machines). The results in Table 2 also suggest that the performance of SIGMA is very satisfactory from the point of view of dependability (only 2 incorrect claims on the "large" dynamic range machine when $N_{\text{SU}} > 3$ and on the "small" dynamic range machine when $N_{\text{SU}} > 4$) 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 $N_{\text{SU}}$ gives a much cheaper method (less function evaluations) at the expense of a loss in effectiveness (greater number of failures).
ACKNOWLEDGMENT: 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].
A NEW METHOD FOR GLOBAL OPTIMIZATION BASED ON STATISTICAL DIFFERENTIAL EQUATIONS (U) CAMERINO UNIV. MATHEMATICS INST. F. ALUFFI-PENTINI ET AL. F/G 12/1 NL
REFERENCES


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 $f(x) = f(x_1, ..., x_N)$ of $N$ real variables $x_1, ..., x_N$. 
2. METHOD

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

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

\[
d\xi = -\nabla f(\xi) dt + \varepsilon(t) dw
\]

with initial condition:

\[
\xi(0) = x_0
\]

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

\[
\xi_{k+1} = \xi_k - h_k \frac{\hat{y}(\xi_k)}{\sqrt{h_k}} u_k + \varepsilon(t_k) \sqrt{h_k}, \quad k = 0, 1, 2, \ldots
\]

\[
\xi_0 = x_0
\]

where \( h_k \) is the time integration steplength, \( \frac{1}{N} \hat{y}(\xi_k) \) is computed as a finite-differences approximation to the directional derivative of \( f \) in a randomly chosen direction, and \( u_k \) is a random sample from an \( N \)-dimensional standard gaussian distribution.

We consider the simultaneous evolution of a number \( N_{\text{TRA}} \) of trajectories during an "observation period" having the duration of a given number \( N_{\text{HP}} \) of time integration steps, and within which the noise coefficient \( \varepsilon(t) \) of each trajectory is kept at a constant value \( \varepsilon_p \), while the steplength \( h_k \) and the spatial increment \( \Delta x_k \) 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 $\varepsilon$ and $\Delta x_k$, and is considered as having the same "past history" of the first.

The number $N_{\text{TRAJ}}$ 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 $N_{\text{PMAX}}$ of observation periods has been reached, or if all the final values of $f(x)$ (except for the discarded trajectory) are equal (within numerical tolerances, and possibly at different points $x$) to their minimum value $f_{\text{TFMIN}}$ ("uniform stop" at the level $f_{\text{TFMIN}}$). In the former case the trial is considered unsuccessful, while in the latter case a comparison is made between the common final function value $f_{\text{TFMIN}}$ and the current best minimum function value $f_{\text{OPT}}$ 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 $f_{\text{OPT}}$.

The trials are repeated with different operating conditions (initial point $x_0$, maximum trial length $N_{\text{PMAX}}$, seed of the noise generator, policy for selecting the starting values for $\varepsilon$ in the second continuation trajectory after branching, and trial-start values for $\varepsilon$) and the complete algorithm is stopped — at the end of a trial — if a given
number $N_{\text{SU}}$ of uniform stops at the current $f_{\text{OPT}}$ level has been obtained, or if a given maximum number $N_{\text{TRIAL}}$ of trials has been reached: success of the algorithm is claimed if at least one uniform stop occurred at the final value of $f_{\text{OPT}}$. 
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, DERCE, RCLOPT, STOOPT, RANGE, INISCA, NOSCA, SEGSCA, VARS, CUMSCA, ACTSCA, MOVSCA, UPDSCA, ALKNUT, GAUSRV, UNITIV; a set of 7 auxiliary functions, IPREC, IPRECE, FUNCT0, ITOLCH, EIGSCA, CHAOS, UNIFRN; and a driver subroutine SIGMA 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, DERCE, FUNCT0, RANGE, NEWH. The value of the function f(x) is computed — whenever required in the numerical integration process — by calling the function FUNCT0. FUNCT0 rescales the variables by calling VARS (see d)), calls RANGE to take care of the cases where the current point x is
outside the admissible range ([1], sect. 3.2.11) calls the user-supplied function FUNCT (sect. 4.5.1) to compute \( f(x) \), and possibly updates the best current function minimum \( f_{\text{OPT}} \) and the corresponding minimizer \( x_{\text{OPT}} \) by calling STOOPT (see c)). The basic step of the numerical integration is performed by SSTEP which calls FUNCT0 to compute the value of \( f(y) \), 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 FUNCT0; 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 \( \Delta x_k \) 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 c... 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 \( \gamma \) (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 $\varepsilon_p$ and for the initial $\Delta x_k$; 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 $b$ ([1], sect. 3.2.12) by calling SEOSCA 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. 4.3.2) using the rescaling capabilities of SEGSQA and VARSQA (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 SIGMA, sect. 4.1).

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 INISQA or NOSQA
actually executes the trial by calling TRIA.
ISTOP = 1 relative difference criterion satisfied
       = 2 absolute difference criterion satisfied
       = 3 both criteria satisfied

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

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

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

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

The subroutine definition statement of PTKSUC is

SUBROUTINE PTKSUC (KSUC)

where

KSUC is the integer variable (1 ≤ KSUC < NSUC)
defined above.

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

A user not interested in the use of any one of the output subroutines must provide the corresponding dummy subroutine (with RETURN as the only executable statement) in order to avoid unresolved references problems.
FOPT is the current best minimum value of \( f \) found from algorithm start (\( f_{OPT} \)) (FOPT is updated whenever a function value \( f(x) \) is computed).

FTMIN, FTMAX are respectively the minimum and the maximum value of \( f(x) \) among the end-of-trial values obtained at the final points of the last trajectories of the current trial (\( f_{TMAX}, f_{TMIN} \)).

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

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

- **ISTOP = 0** The maximum number NPMAX of observation periods has been reached.
- **ISTOP ≠ 0** all the final values of \( f(x) \) of the last observation period (except for the just discarded trajectory) are close enough to their common minimum value FPFMIN, with respect to an absolute or relative difference criterion, ([1], sect. 3.2.13), to be considered numerically equal.

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

The absolute value indicates which of the difference criteria was satisfied.
taken place, if NSUC (input parameter to SIGMA) had been given a (lower) value, equal to the current KSUC.

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

The subroutine definition statement of PTSEG is

```plaintext
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 \( f(x) \) 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

```plaintext
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 (\( x_{OPT} \)).
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 > 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 \( f(x) \) (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 \( x \) of the function \( f \) to be minimized.

The function definition statement is

\[
\text{DOUBLE PRECISION FUNCTION FUNCT (N,X)}
\]

where

\( N \) is the (input) dimension of the problem
\( X \) is the (input) \( N \)-vector containing the coordinates of the point \( x \) where the value of \( f \) is to be computed.

Note that the function \( f(x) \) 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 SIGMA

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 SIGMA is included in the package. SIGMA simply calls SIGMA after assigning default values to a number of input parameters. The subroutine definition statement is:

SUBROUTINE SIGMA (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 SIGMA as follows:

\[ H = 10^{-10} \]
\[ EPS = 1 \]
\[ DX = 10^{-9} \]
\[ IRAND = 0 \]
\[ NTRAJ = 0 \]
\[ ISEGBR = 0 \]
\[ KPBR0 = 0 \]
\[ INKPBR = 0 \]
\[ NPMIN = 10 \]
\[ NPMAX0 = 100 \]
\[ INPMAX = 50 \]
\[ NTRIAL = \max(50, 5 \times NSUC) \]
\[ TOLREL = 10^{-3} \]
\[ TOLABS = 10^{-6} \]
\[ KPASCA = 10 \text{ (if } N \leq 5) ; = 300 \text{ (if } N > 5) \]
\[ INHP = 1 \]
\[ XMIN(I) = -10^4 \text{ (I = 1, \ldots, N)} \]
\[ XMAX(I) = 10^4 \text{ (I = 1, \ldots, N)} \]
NPIMIN (say $5 < \text{NPIMIN} < 100$)

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 untolerable growth of the computation effort or an unacceptable degradation of the performance, the user is advised to modify (if needed) the above parameters starting from NSUC, based on information from the output subroutines (PTRIAL and PTKSUC). Note that NSUC is the only "free" control parameter of the driver SIGMA1 (Sect. 4.4).

The value of KPASCA should be based on possible analytical or experimental evidence on the ill-scaling of the function $f(x)$. 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 ($\text{XRMIN}$, $\text{XRMAX}$) 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 (includ-
ing those used for the computation of derivatives, and for
the rejected time-integration steps).

IOUT is the (output) indicator of the stopping conditions as follows:
If IOUT = -99 a fatal error was detected when performing some preliminary
checking of the input data, and the algorithm was not even started; otherwise
the algorithm was started, and the value of IOUT is the final value of the
of the parameter ISTOPT (an output indicator of the output subroutine
PTRIAL, described in sect. 4.5.2.).

Success is claimed by the algorithm it IOUT > 0, i.e. if at least
one of the trials stopped with a positive value of the trial stopping indi-
cator 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, KPBRO (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 $R_{MIN}^i$, $R_{MAX}^i$).

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

IRAND is a control (input) index for the initialization of the random number generator:
if IRAND > 0 the generator is initialized before starting the trial $K_t$ with seed IRAND + $K_{t-1}$;
if IRAND <= 0 the generator is initialized (with seed 0) only at the first call of SIGMA.

INHP is used to control the number NHP ("duration") of time integration steps for observation period $K_p$ as follows:
if INHP = 1, NHP = 1 + \lfloor \log_2(K_p) \rfloor, ("short" duration)
if INHP = 2, NHP = \lfloor \sqrt{K_p} \rfloor, ("medium" duration)
if INHP = 3, NHP = K_p, ("long" duration),
where \lfloor x \rfloor 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 \((\text{NTRAJ} = 7)\), and if the input value is outside the interval \((3, 20)\) \(\text{NTRAJ}\) is set to the nearest extreme value).

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

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

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

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

\(\text{NSUC}\) is the (input) number of successful trials (with the same final value \(f_{\text{OPT}}\), see sect. 2) after which the computation is stopped \((\text{NSUC})\).

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

\(\text{TOLREL}\) and \(\text{TOLABS}\) are the (input) relative and absolute tolerances for stopping a single trial \((\tau_{\text{REL}}, \tau_{\text{ABS}}\), see [1], sect. 3.2.8).
SIGMA returns to the calling program the output parameters
NTRAJ, ISEGBR, KPBR, 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, FTPMIN, FTPMAX, FTPOPT
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 x)
XØ is an N-vector containing the initial values of the x-variables
H is the initial value of the time integration steplength.
EPS is the initial value of the noise coefficient
DX initial value of the magnitude of the discretization increment (Δx) for computing the finite-differences derivatives.
NTRAJ is the number of simultaneous trajectory segments (NTRAJ).
(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 para-
meters 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

\[
\text{CALL SIGMA (N, X\&0, H, EPS, DX,}
\]

\[
\text{NTRAJ, ISEGMBR, KPBR\&0, INKPBR,}
\]

\[
\text{NPMIN, NPMAX\&0, INPMAX,}
\]

\[
\text{NSUC, NTRIAL, TOLREL, TOLABS, XRMIN, XRMAX,}
\]

\[
\text{KPASCA, IRAND, INHP, IPRINT,}
\]

\[
\text{XMIN, FMIN, NFEV, IOUT)}
\]

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 (-1,1), 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 $\mathbb{R}^N$. 
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 STOLOPT and RCLOPT respectively "store" and "recall"
the current values of the best minimum POPT and of the corresponding
minimizer XOPT.

d) Subprograms for rescaling the variables: INISCA, NOSCA, SEGSCA,
VARSQA, 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.
VARSQA computes the rescaled variables $Ax + b$.
CUMSCA stores cumulated statistical data on the ill-conditioning
of $f(Ax + b)$.
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 $b$ by calling
EIGSCA and VARSQA.
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 SIMA 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 1000 (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 $N = 2$, $x = (x_1, x_2)^T$, and consider the six-humps camel function $f(x) = \frac{1}{3} x_1^6 - 2.1 x_1^4 + 4 x_1^2 + x_1 x_2 + 4 x_2^4 - 4 x_2^2$ which has four non-global minima, and two global minima at $x \approx \pm (0.089842, 0.71266)^T$ where $f \approx -1.0313$. The sample program listed in fig. 1, which uses the easy-to-use driver SIMA, was run on a UNIVAC 1100/82 computer with EXEC8 operating system (level 38R5) and ASCII FORTRAN compiler (version 10R1A), starting from $x_0 = (0, 0)^T$ 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 $NSUC$ had been equal to 1 (resp. 2), the minimum would have been found with only 3697 (resp. 8545) function evaluations.

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

DOUBLE PRECISION FUNCTION FUNCT (N,)
C COMPUTES THE VALUE AT X OF THE SIX-HUMP CAMEL FUNCTION
C
DOUBLE PRECISION X,XX,YY

XX = X(T)*X(1)
YY = X(T)*X(2)
FUNCT = ((XX/3.0-2.100)*YY+4.00)*XX+X(1)*X(2)
* +4.00*(YY-1.00)*YY
RETURN
END

Fig. 1 -- List of the Sample Program
REFERENCES


APPENDIX A6

The FORTRAN package SIGMA.
SUBROUTINE SIGMA1(N,XO,NSUC,IPRINT,XMIN,FMIN,NFEV,IOUT)

C SIGMA1 IS A "DRIVER" SUBROUTINE WHICH SIMPLY CALLS THE PRINCIPAL
C SUBROUTINE SIGMA AFTER HAVING ASSIGNED DEFAULT VALUES TO A NUMBER
C OF INPUT PARAMETERS OF SIGMA, AND WAS THEREFORE CONSIDERABLE
C EASIER TO CALL SIGMA1 INSTEAD OF SIGMA, THUS AVOIDING THE
C TROUBLE OF ASSIGNING A VALUE TO ALL THE INPUT PARAMETERS OF SIGMA.
C ALL THE PARAMETERS IN THE DEFINITION OF SIGMA HAVE THE SAME MEANING
C AS IN SIGMA.

C THE USER OF SIGMA1 MUST ONLY GIVE VALUES TO THE INPUT PARAMETERS
C N, XO, NSUC, IPRINT
C AND OBTAINS ON OUTPUT THE SAME OUTPUT PARAMETERS OF SIGMA
C XMIN, FMIN, NFEV, IOUT

C WE RECALL HERE THE MEANING OF THE ABOVE PARAMETERS

C N IS THE PROBLEM DIMENSION (NUMBER OF VARIABLES)
C XO IS AN N-VARIABLE CONTAINING THE INITIAL VALUES OF THE
C NSUC IS THE NUMBER OF SUCCESSFUL TRIALS (WITH THE SAME FINAL
C VALUE OF FOPT) AFTER WHICH THE COMPUTATION IS STOPPED.
C IPRINT IS AN INDEX USED TO CONTROL THE AMOUNT OF PRINTED OUTPUT
C BY CONTROLLING THE CALLS TO THE USER-SUPPLIED OUTPUT SUB-
C ROUTINES PSEG (END-OF-SEGMENT OUTPUT), PTRIAL (END-
C OF-TRIAL OUTPUT), AND PTKSUC (END-OF-TRIAL OUTPUT RELATED
C TO THE COUNT OF SUCCESSFUL TRIALS), WHICH ARE DESCRIBED
C BELOW.
C IPRINT.LE.0 NO CALL TO THE OUTPUT SUBROUTINES
C IPRINT.EQ.0 CALL ONLY PTRIAL AND PTKSUC
C IPRINT.GT.0 CALL ALL OUTPUT SUBROUTINES.
C XMIN IS AN N-VARIABLE CONTAINING THE COORDINATES OF THE POINT
C (OR POSSIBLY ONE OF THE POINTS) WHERE THE FINAL VALUE FMIN
C OF FOPT WAS FOUND.
C FMIN IS THE FINAL VALUE OF THE BEST CURRENT MINIMUM FUNCTION VALUE
C FOPT IS THE TOTAL NUMBER OF FUNCTION EVALUATION (INCLUDING
C THOSE USED FOR THE COMPUTATION OF DERIVATIVES, AND ERR
C THE REJECTED TIME-INTEGRATION STEPS).
C IOUT IS THE INDICATOR OF THE STOPPING CONDITIONS, AS FOLLOWS
C IF IOUT = -99 A FATAL ERROR WAS DETECTED WHEN PERFORM-
C ING SOME PRELIMINARY CHECKING OF THE INPUT DATA, AND
C THE ALGORITHM WAS NOT EVEN STARTED
C OTHERWISE THE ALGORITHM WAS STARTED, AND THE VALUE OF
C IOUT IS THE FINAL VALUE OF THE INTERNAL PARAMETER ISTOP
C (AN OUTPUT INDICATOR OF THE USER-SUPPLIED SUBROUTINE
C PTRIAL).
C SUCCESS IS CLAIMED BY THE ALGORITHM IF IOUT .GT. 0,
C I.E. IF AT LEAST ONE OF THE TRIALS STOPPED UNIFORMLY AT THE
C LEVEL OF THE CURRENT FOPT.

DOUBLE PRECISION XO,XMIN,FMIN
DOUBLE PRECISION DX, EPS, H, TOLABS, TOLBEL
DOUBLE PRECISION VRMIN, VRMAX, XRMIN, XRMAX
DIMENSION XRMIN(100), XRMAX(100)
DATA VRMIN, VRMAX / -1.04, 1.04/
DATA NTRIAL/50/

M = 1.0E-10
EPS = 1.0E-9
DX = 1.0E-9
IRAND = 0
NTRAJ = 0
ISEGBR = 0
INKPBR = 0
KPBRO = 0
NPMIN = 10
NPMA0 = 100
INPMAX = 50
NTRIAL = MAX(0, NTRIAL, 5 + MSUC)
TOLREL = 1.0E-3
TOLABS = 1.0E-6
KPASCA = 10
IF(N.GT.5)KPASCA = 300
INHP = 1
DO 1 IX = 1, N
  XRMIN(IX) = VRMIN
  XRMAX(IX) = VRMAX
  CONTINUE
1 1
CALL SIGMA (XO, EPS, H, DX,
2 NTRAJ, ISEGEB, KPBR0, INKPBR,
3 NPBR0, NPMA0, INPMAX,
4 NSUC, NTRIAL, TOLREL, TOLABS, XRMIN, XRMAX,
5 NPASCA, IRAND, INHP, IPRINT,
6 XRMIN, FRMIN, NFEV, IOUT )
C THE SUBROUTINE SIGMA IS THE PRINCIPAL SUBROUTINE OF THE PACKAGE
C SIGMA, WHICH ATTEMPTS TO FIND A GLOBAL MINIMIZER OF A REAL VALUED
C FUNCTION F(X) = f(x1, ..., xn) OF n REAL VARIABLES x1, ..., xn.
C THE ALGORITHM AND THE PACKAGE ARE DESCRIBED IN DETAIL IN THE TWI
A GLOBAL MINIMIZATION ALGORITHM USING STOCHASTIC DIFFERENTIAL EQUATIONS

BY STATISTICAL MECHANICS, STARTING FROM AN INITIAL POINT X₀,

X IS UPDATED BY THE (STOCHASTIC) DISCRETIZATION STEPS

WHERE DX₁ = - ∂f/∂x + U (FIRST HALF-STEP)

DX₂ = EPS * SQRT(H) * U (SECOND HALF-STEP)

H IS THE TIME-INTEGRATION STEPLENGTH.

G∗/H IS COMPUTED AS A FINITE-DIFFERENCE APPROXIMATION TO THE

DIRECTIONAL DERIVATIVE OF f ALONG AN ISOTROPICALLY RANDOM 1-DIRECTION.

EPS IS A POSITIVE "NOISE" COEFFICIENT, AND

U IS A RANDOM SAMPLE FROM AN M-DIMENSIONAL GAUSSIAN DISTRIBUTION.

WE CONSIDER THE SIMULTANEOUS EVOLUTION OF A GIVEN FIXED NUMBER

NTRAJ OF TRAJECTORIES DURING AN OBSERVATION PERIOD IN WHICH FOR

EACH TRAJECTORY EPS IS FIXED WHILE H THE SPATIAL DISCRETIZATION INCREMENTS D^ι FOR COMPUTING G∗ ARE AUTOMATICALLY

ADJUSTED BY THE ALGORITHM.

AFTER EVERY OBSERVATION PERIOD ONE OF THE TRAJECTORIES IS DISCARDED,

ALL OTHER TRAJECTORIES CONTINUE UNPERTURBED, AND ONE OF THEM IS SE-...
The call statement is

\texttt{CALL SIGMA ( N, XO, H, EPS, DX, NTRAJ, IS6BR, KPBRO, INKPB, NMIN, NMAXO, IMPMAX, NSUC, NTRIAL, TOLREL, TOLABS, XMIN, XMAX, KPBRO, IRAND, INHP, IPRINT, XMIN, FMIN, MFV, IOUT )}

Call parameters

Input parameters are those in lines 1, 3, 4, 5 of the call statement.

Input-output parameters are those in line 2.

Output parameters are those in line 6.

Note that a number of other (internal) parameters can be obtained by means of the user-supplied output subroutines \texttt{PMES}, \texttt{PRUAL}, \texttt{PIKCB}, which are described below.

Description of the call parameters

\texttt{C N IS THE PROBLEM DIMENSION (NUMBER OF VARIABLES)}
\texttt{C XO IS AN N-VECTOR CONTAINING THE INITIAL VALUES OF THE}
\texttt{C VARIABLES}
\texttt{C H IS THE INITIAL VALUE OF THE TIME-INTEGRATION STEP LENGTH}
\texttt{C EPS IS THE INITIAL VALUE OF THE NOISE COEFFICIENT}
\texttt{C DX IS THE INITIAL VALUE OF THE MAGNITUDE OF THE DISCRETIZATION}
\texttt{C INCREMENT FOR COMPUTING THE FINITE-DIFFERENCE DERIVATIVES}
\texttt{C NTRAJ IS THE NUMBER OF SIMULTANEOUS TRAJECTORIES}
\texttt{C (NOTE HOWEVER THAT IF THE INPUT VALUE IS ZERO, NTRAJ IS}
\texttt{C SET TO A DEFAULT VALUE (NTRAJ = 9), AND IF THE INPUT VALUE}
\texttt{C IS OTHERWISE OUTSIDE THE INTERVAL (3, 20) NTRAJ IS SET TO}
\texttt{C THE NEAREST EXTREME VALUE)}
\texttt{C IS6BR, KPBRO, INKPB DETERMINE, AT THE END OF AN OBSERVATION}
\texttt{C PERIODS, WHICH ONE OF THE SIMULTANEOUS TRAJECTORIES}
\texttt{C IS TO BE BRANCHED, AS FOLLOWS}
\texttt{C BRANCHING IS NORMALLY PERFORMED ON THE TRAJECTORY WHICH}
\texttt{C OCCUPIES THE PLACE IS6BR IN THE TRAJECTORY SELECTION ORDERING, EXCEPT AT (THE END OF) EXCEPTIONAL OBSERVATION}
\texttt{C PERIODS, WHERE THE FIRST TRAJECTORY IN THE ORDERING IS}
\texttt{C BRANCHED. EXCEPTIONAL BRANCHING OCCURS AT THE OBSERVATION}
\texttt{C PERIODS NUMBERED KP = KPBRO \* J, INKPB, (J = 1, 2, 3, ...).}
\texttt{C THEREFORE IS6BR SELECTS THE LEVEL (IN THE ORDERING) AT}
\texttt{C WHICH NORMAL BRANCHING OCCURS, WHILE KPBRO AND INKPB}
\texttt{C SELECT THE FIRST OCCURRENCE AND THE REPEITION FREQUENCY}
\texttt{C OF THE EXCEPTIONAL OBSERVATION PERIODS}
\texttt{C (NOTE HOWEVER THAT IF ONE OF THE INPUT VALUES IS ZERO,}
\texttt{C THE CORRESPONDING VARIABLE IS SET TO A DEFAULT VALUE}
\texttt{C IS6BR = INT(1 + NTRAJ)/2, INKPB = 10, KPBRO = 3,}
\texttt{C IF THE INPUT VALUE FOR IS6BR IS OTHERWISE OUTSIDE THE}
\texttt{C INTERVAL (1, NTRAJ), IS6BR IS SET TO THE NEAREST}
\texttt{C EXTREME VALUE, AND IF KPBRO HAS A VALUE NOT INSIDE THE}
\texttt{C INTERVAL (1, INKPB), IT IS ASSIGNED THE SAME VALUE}
\texttt{C MODULO INKPB)}
\texttt{C NMIN IS THE MINIMUM DURATION OF A TRIAL, I.E. THE MINIMUM}
\texttt{C NUMBER OF OBSERVATION PERIODS THAT SHOULD ELAPSE BEFORE}
A SUCCESS IS CLAIMED FOR THE FIRST TRIAL OF MAXIMUM ACCEPTABLE NUMBER OF OBSERVATION PERIODS IN A TRIAL.

The number of successful trials (with the same final value) after which the computation is stopped.

TOLREL AND TOLABS ARE THE RELATIVE AND ABSOLUTE TOLERANCES.

FOR STOPPING A SINGLE TRIAL.

XMIN, XRMAX ARE N-VECTORS DEFINING THE ADMISSIBLE REGION FOR THE X-VALUES, WITHIN WHICH THE FUNCTION VALUES CAN BE SAFELY COMPUTED.

XPASCA IS THE MINIMUM NUMBER OF TRAJECTORY SEGMENTS THAT SHOULD BE ELAPSED BEFORE THE RESCALING PROCEDURES ARE ACTIVATED.

IRAND IS A CONTROL INDEX FOR THE INITIALIZATION OF THE RANDOM NUMBER GENERATOR.

IRAND.GT.0 THE GENERATOR IS INITIALIZED, BEFORE STARTING THE TRIAL KT, WITH SEED IRAND.KT-1.

IRAND.LE.0 THE GENERATOR IS INITIALIZED WITH SEED 0, ONLY AT THE FIRST CALL OF SIGMA.

INHP IS A CONTROL INDEX FOR SELECTING THE NUMBER NHP OF TIME-INTEGRATION STEPS FOR OBSERVATION PERIOD KP (DURATION IF TRAIL KP AS FOLLOWS (LOG Is BASE 2).

INHP=1 NHP = INT(4(KP)) ("SHORT" DURATION)
INHP=2 NHP = INT(3(KP)) ("MEDIUM" DURATION)
INHP=3 NHP = KP ("LONG" DURATION)

IPRINT IS AN 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), WHICH ARE DESCRIBED BELOW.

IPRINT.LT.0 NO CALL TO THE OUTPUT SUBROUTINES.
IPRINT.EQ.0 CALL ONLY PTRIAL AND PTKSUC.
IPRINT.GT.0 CALL ALL OUTPUT SUBROUTINES.

XMIN IS AN N-VECTORS CONTAINING THE COORDINATES OF THE POINT OR POSSIBLY ONE OF THE POINTS WHERE THE FINAL VALUE XMIN OF FOPT WAS FOUND.

XMIN IS THE FINAL VALUE OF THE BEST CURRENT MINIMUM FUNCTION VALUE FOPT.

NFEV IS THE TOTAL NUMBER OF FUNCTION EVALUATION (INCLUDING THOSE USED FOR THE COMPUTATION OF DERIVATIVES, AND FOR THE REJECTED TIME-INTEGRATION STEPS).

IOUT IS THE INDICATOR OF THE STOPPING CONDITIONS, AS FOLLOWS:
IOUT = -99 A FAULTY ERROR WAS DETECTED WHEN PERFORMING SOME PRELIMINARY CHECKING OF THE INPUT DATA, AND THE ALGORITHM WAS NOT STARTED.
IOUT IS THE FINAL VALUE OF THE INTERNAL PARAMETER ISTOP.
IOUT IS THE FINAL VALUE OF THE USER-SUPPLIED SUBROUTINE PTRIAL, DESCRIBED BELOW.
SUCCESS IS CLAIMED BY THE ALGORITHM IF IOUT .GE. 0.
I.E. IF AT LEAST ONE OF THE TRIALS STOPPED WITH A POSITIVE VALUE OF THE TRIAL STOPPING INDICATOR ISTOP (AN OUTPUT INDICATOR OF THE USER-SUPPLIED SUBROUTINE PTRIAL, DESCRIBED BELOW), AND NO LOWER VALUE FOR FOPT WAS FOUND.
IN THE FOLLOWING TRIALS,

USER-SUPPLIED SUBPROGRAMS

THE USER MUST PROVIDE THE FUNCTION FUNCT TO COMPUTE F(X),
AND THE THREE OUTPUT SUBROUTINES PTSEG, PTrial, PTriRAL.
THE CALLS TO THE OUTPUT SUBROUTINES ARE CONTROLLED BY IPRINT
(INPUT PARAMETER TO SIGNAL).

A USER NOT INTERESTED IN USING ANY ONE OF THE OUTPUT SUBROUTINES
MUST PROVIDE A DUMMY SUBROUTINE (WITH RETURN AS THE ONLY
EXECUTABLE STATEMENT) TO AVOID UNRESOLVED REFERENCES.

IN THE FOLLOWING DESCRIPTION ALL NON-INTEGER ARGUMENTS ARE
DOUBLE PRECISION (INTEGER ARGUMENTS ARE INDICATED AS
EXECUTABLE STATEMENT)

THE FUNCTION FUNCT

MUST RETURN AS ITS VALUE THE VALUE AT X OF THE FUNCTION
TO BE MINIMIZED

WHERE

W IS THE (INPUT) DIMENSION OF THE PROBLEM.
X IS THE (INPUT) N-VECTOR CONTAINING THE COORDINATES OF THE
POINT X WHERE THE FUNCTION IS TO BE COMPUTED.

THE SUBROUTINE PTSEG

PTSEG IS CALLED (IF IPRINT.GT.0) AT THE END OF EVERY OBSER-
VATION PERIOD.

WHERE

N IS THE (INPUT) DIMENSION OF THE PROBLEM.
FPFIN, FPFIN ARE RESPECTIVELY THE MINIMUM AND THE MAXIMUM
AMONG THE VALUES OF F(X) OBTAINED AT THE FINAL POINTS OF
TRAJECTORY SEGMENTS OF THE (JUST ELAPSED) OBSERVATION
PERIOD KP.

XPFIN IS AN N-VECTOR CONTAINING THE COORDINATES OF THE
(FINAL) POINT OR POSSIBLY ONE OF THE POINTS WHERE
FPFIN WAS OBTAINED.

KP IS THE TOTAL NUMBER OF ELAPSED OBSERVATION PERIODS SIN-
CURRENT TRIAL.
NFEV IS THE TOTAL NUMBER OF FUNCTION EVALUATIONS PERFORMED
FROM ALGORITHM START.

THE SUBROUTINE P Trial

P Trial IS CALLED (IF IPRINT.GE.0) AT THE END OF EVERY TRIAL.

WHERE

N IS THE (INPUT) DIMENSION OF THE PROBLEM.
XOPT IS AN N-VECTOR CONTAINING THE COORDINATES OF THE
POINT (OR POSSIBLY ONE OF THE POINTS) WHERE THE CURRENT
MINIMUM F0PT WAS OBTAINED.
F0PT IS THE CURRENT BEST MINIMUM VALUE FOUND FOR F FROM
ALGORITHM START (FOPT IS UPDATED WHENEVER A FUNCTION
VALUE IS COMPUTED).
FITMIN, FITMAX ARE RESPECTIVELY THE MINIMUM AND THE MAXIMUM
AMONG THE VALUES OF F(X) OBTAINED AT THE FINAL POINTS OF
THE LAST TRAJECTORY SEGMENTS OF THE CURRENT TRIAL.
FITFOPT IS THE CURRENT MINIMUM VALUE OF FITMIN AMONG THE
TRIALS WHICH DID NOT STOP FOR REACHING THE MAXIMUM ALLOWED
NUMBER OF SEGMENTS (STOPPING INDICATOR ISTOP = 0; SEE
BELOW). FITFOPT IS USED BY SIGMA TO COMPUTE WEIGHTS, THE
PARAMETER TO THE OUTPUT SUBROUTINE PMSUC (SEE BELOW).
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.
ISTOP IS THE INDICATOR OF THE STOPPING CONDITION OF THE TRIAL,
AS FOLLOWS
ISTOP = 0
THE MAXIMUM NUMBER NMAX OF OBSERVATION PERIODS HAS
BEEN REACHED.
ISTOP = N
ALL THE END-OF-SEGMENT VALUES OF F(X), EXCEPT FOR THE
JUST DISCARDED SEGMENT) ARE CLOSE ENOUGH TO THEIR COMMON
MINIMUM VALUE FITMIN, WITH RESPECT TO AN ABSOLUTE OR
RELATIVE DIFFERENCE CRITERION, TO BE CONSIDERED NUMERI-
CALLY EQUAL.
THE ABSOLUTE VALUE AND THE SIGN OF ISTOP HAVE THE
FOLLOWING MEANING:
THE ABSOLUTE VALUE INDICATES WHICH DIFFERENCE
CRITERION WAS SATISFIED
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 FITMIN AND THE CURRENT
BEST MINIMUM VALUE FOPT (WHICH IS UPDATED WHEN-
EVER A FUNCTION VALUE IS COMPUTED)
ISTOP = 0
FITMIN IS NUMERICALLY EQUAL (W.R.T. AT LEAST
ONE OF THE ABOVE DIFFERENCE CRITERIA) TO FOPT
ISTOP = 0
FITMIN IS NOT EVEN NUMERICALLY EQUAL TO FOPT
AND THEREFORE CANNOT BE CONSIDERED AS AN
ACCEPTABLE GLOBAL MINIMUM.
ISTOP IS THE VALUE OF THE TRIAL STOPPING INDICATOR ISTOP
CORRESPONDING TO THE (CURRENT OR PAST) TRIAL WHERE FITFOPT
WAS OBTAINED, WITH THE SIGN WHICH IS UPDATED ACCORDING TO
THE COMPARISON BETWEEN FITFOPT AND THE PRESENT VALUE OF
FOPT, AS DESCRIBED ABOVE.
THE FINAL VALUE OF ISTOP IS RETURNED BY SIGMA AS THE VALUE
OF THE OUTPUT INDICATOR IOUT OF THE ALGORITHM STOPPING CON-
DITIONS (WHENEVER THE ALGORITHM WAS STARTED, IOUT.NE.—99,
SEE ABOVE).
THE SUBROUTINE PMSUC
PTKSUC is called only at the end of every successful trial such that an increment occurred in the value KSCU of the maximum number of successful trials at the same (current or past) value of F0BY. A call to PTKSUC therefore provides the user with the operationally interesting information that a final success claim would have taken place, if NSUC (input parameter to SIGMA) had been given a lower value, equal to the current KSUC.

PTKSUC is called only if IPRINT .GE. 0 and KSUC.LT.NSUC.

The definition statement is subroutine PTKSUC (KSUC).

Where KSUC is the integer variable (1.LE.KSUC.LT.NSUC) defined above.

DOUBLE PRECISION X0, W, EPS, DX, TOLREL, TOLABS
DOUBLE PRECISION XRMIN, XRMAX, XMIN, FMIN
DOUBLE PRECISION EPSAG, EPSAP, EPSC
DOUBLE PRECISION EPSMAX, EPSR, F, FOPT, FTMAX
DOUBLE PRECISION XH, XE, XMV, EPSCO, WMX, VR, X
DOUBLE PRECISION XMIC, XMVC, XOPT, FOPTC
DOUBLE PRECISION EPS, BIAS, EPSA, EPSAMAX

DIMENSION X0N), XRMIN(N), XRMAX(N)

COMMON /DIMCOM/ X(100, 20), XH(20), XRMIN(20), XRMAX(20), XRMIN(20), XRMAX(20)
COMMON /SECOM/ FMIN, EPSAP, EPSAMAX

DATA FOR THE VARIATION OF NOISE COEFFICIENT

DATA EPSR1, EPSAMAX, EPSA1

INITIALIZE COMMON AREA /DIMCOM/

CALL INIT(N, XO, H, ERS, DX, TRAND, F
1 NRRAJ, IS66GR, K, PBR0, K, PBR1, K, PBR0, K, PBR1, K, PBR0, K, PBR1, K, PBR0, K, PBR1, K, PBR0, K, PBR1
CHECK PARAMETER VALUES

IF (NPMAX.LT.0. OR. NPMAX.GT.0. OR. NTRIAL.LT.0. OR. NTRIAL.GT.0.) RETURN

IF (IOUT .EQ. -99) RETURN

INITIALIZE VARIABLES

EPSC = EPS

436. C
437. C
438. C
439. C
440. C
441. C
442. C
443. C
444. C
445. C
446. C
447. NPMAX = NPMAX0
448. ISTOPT = 0
449. ISTOP = 0
450. ICCON = (NTrial+NTrial'4)/5
451. NFEV = 0
452. NTES = NSUC
453. ICTS = NSUC-1
454. FTOPT = F
455. C
456. C START SERIES OF TRIALS
457. C DO 30 IC = 1,NTrial
458. C
459. C SET INITIALIZATION INDEX FOR NOISE GENERATOR
460. C
461. IS = 0
462. IF(RAND.GT.0.01)IS = IRAND+IC-1
463. C
464. C INITIALIZE TRIAL
465. C IF(IC.GT.T. AND IC.LE.ICCON) CALL REINIT(N,X0,EPSC,IS,F,IFEF)
466. C IF(C.GT.T. AND IC.LE.ICCON) CALL REINIT(N,XMIN,EPSC,IS,FOP3,IFEF)
467. C FTFMN = F
468. C FTMAX = F
469. C NFEV = NFEV+1
470. C
471. C PRINT INITIAL CONDITIONS OF TRIAL
472. C IF(IPRINT.GT.0) CALL PISEG(N,X0,FTFMN,FTMAX,0,NFEV)
473. C
474. C DEACTIVATE SCALING
475. C IF(KPASCA.GT.1+NPMAX.NOR.W.LE.1) CALL NOSCA
476. C
477. C INITIALIZE COMMON AREA /SCALE/
478. C IF(KPASCA.LE.NPMAX.AND.W.GT.1) CALL INISCA(N,NTRAJ)
479. C
480. C PERFORM A TRIAL
481. C CALL TRIAL(N,XMIN,NPMAX,KPASCA,TOLREL,TOLABS,1)
482. C IF(IPRINT,EQ.0) CALL PRINT(N,XMIN,FTFMN,FTMAX,NFEV,0,ISTOP)
483. C
484. C EVALUATE PAST TRIAL AND PREPARE NEXT TRIAL
485. C
486. C SET TRIAL DURATION
487. C IF(ISTOP.EQ.0) NPMAX = NPMAX+NPMAX
488. C
489. C RESET CURRENT NUMBER OF SUCCESSES REQUIRED BEFORE STOPPING
490. C COMPUTE INDICATOR OF TRIAL STOPPING CONDITIONS
491. C UPDATE BEST CURRENT VALUES OF TRIAL STOPPING INDICATOR AND
C OF FUNCTION F(X) AT TRIAL STOP

IF((FTFMN.GT.FTOPT).OR.(ISTOP.EQ.0).AND.(ISTOPT.NE.0))GO TO 10
IF((ITOLCH(FTFOPT,FTFMN,TOLRELTOLABS).EQ.0))NTES = NSUC

FITFOPT = FTFMN
ISTOPT = ISTOP
CONTINUE
ISTOPT = IABS(ISTOPT)
CALL HCLOPT(N,XMIN,FOPT)
IF((ITOLCH(FTFOPT,FOPT,TOLRELTOLABS).EQ.0))ISTOPT = -ISTOPT
IF((ITOLCH(FTFMN,FOPT,TOLRELTOLABS).EQ.0))ISTOP = -ISTOP
C END-OF-TRIAL PRINT OUT
IF(IPRINT.GE.0)
1 CALL PTRIAL (N,XOPT,FOPT,FTFMN,FTFMX,FTFOPT,ISTOPT,ISTOPT, NFEV, KP, IPRINT)
C UPDATE INITIAL VALUE OF NOISE COEFFICIENT FOR NEXT TRIAL
C
IF((ISTOP.EQ.0).AND.(FTFOPT.LE.0.0))IFEP = 1
IF((ISTOPT.NE.0).AND.(FROPT.LT.0.0))IFEP = 2
IF(IPRINT.GE.0)
C UPDATE, PRINT, AND CHECK TRIAL STOPPING CONDITIONS
C
IF((ISTOPT.GT.0.0).AND.(ISTOPT.GT.0.0))NTES = NTES-1
IF((NTES.LT.0.0).OR.(ISTOPT.LT.0.0).OR.(ISTOPT.GT.0.0).OR.(NTES.GT.0.0).OR."
1" IPRINT.LT.0) Go TO 20
KTES = NSUC-ICTS
CALL PTKSUC(KSU,C)
ICTS = NTES-1
CONTINUE
IF((NTES.LT.0.0).AND.(ISTOPT.GT.0.0).AND.(ISTOPT.GT.0.0))GO TO 40
C CONSTRAIN NOISE COEFFICIENT WITHIN BOUNDS
C
EPC = DMRT(EPC,EPSMAX)
IF((EPC.LE.0.0).OR.(EPC.GT.0.0))CONTINUE
50 CONTINUE
C END OF SERIES OF TRIALS
C
CONTINUE
C INDICATOR OF STOPPING CONDITIONS
C
IOUT = ISTOPT
CALL DERECM(NDIM,X,FV,DX,W,DFDX)

C TRY AGAIN THE FIRST HALF-STEP
DO 40 IC = 1,NDIM
1 XP(IC) = X(IC) - H*W(IC) + DFBX + DBLE(FLOAT(NDIM))
1 CONTINUE
F = FUNCT0(NDIM,XP)
FVS = FV + DX + DABS(DEDXV - DFDX)

C UPDATE STEP LENGTH H AND ACCEPT OR REJECT THE FIRST HALF-STEP
C
CALL NEWH(NTIM,FVS,F,X,E,IE)

C UPDATE CUMULATED PAST SCALING DATA
IF(IEC. GE. 1) CALL CUMSCA(NDIM,W,DFDX)
IF (IEC. GE. 1) GO TO 10
DO 9 IC = 1,NDIM
1 X(IC) = P(IC)
9 CONTINUE
O = X

C UPDATE CUMULATED PAST SCALING DATA
CALL CUMSCA(NDIM,W,DFDX)
IF (TOC(W,FV,FS,TOA,J,ABS).EQ.0) DX = DX/RCD
IF (TOC(W,FV,FS,TOA,J,ABS).GT.0) DX = DX*RCD
EPSR = DSQRT(H)*EPS
C TAKE A SAMPLE INCREMENT OF THE WIENER PROCESS
CALL GausRV(NDIM,W)
C TRY THE SECOND HALF-STEP
DO 40 IC = 1,NDIM
1 XP(IC) = XP(IC) + EPSR*W(IC)
40 CONTINUE
F = FUNCT0(NDIM,XP)
41 C ACCEPT OR REJECT THE COMPLETE STEP
IF (F - FV. LE. EPS*EPS*SF) GO TO 70
H = H + HR
IE = IE + 1
IF (H. GT. NMINS) GO TO 20
C
C STEP ACCEPTED
DO 50 IC = 1,NDIM
1 X(IC) = XP(IC)
50 CONTINUE
DX = DMIN1(DX,DXMAX)
5 DX = DMAX1(DX,DXMIN)
C
C RETURN
RETURN
END
COMPUTED
CALLS THE SUBROUTINE DERFOR (OR DERCEN ) TO COMPUTE THE FORWARD -
(C OR CENTRAL -) DIFFERENCES DIRECTIONAL DERIVATIVE 6AM/H
CALLS THE SUBROUTINE NEWM TO ACCEPT OR REJECT THE FIRST HALF -
STEP AND OBTAIN AN UPDATED VALUE FOR H
CALLS THE SUBROUTINE EUMSCA TO UPDATE THE CUMULATED SCALING DATA
UPDATES THE SPATIAL DISCRETIZATION INCREMENT DIX BASED ON THE
RESULTS OF CALLING THE FUNCTION ITOLCH
CALLS THE SUBROUTINE GAUSRV TO PERFORM THE SECOND HALF-STEP.

DOUBLE PRECISION XH, EPS, DXF
DOUBLE PRECISION DDX, DFBX, DXMAX, DXMIN
DOUBLE PRECISION EPSR, F5, FV, FS, HMIN, HR, HS
DOUBLE PRECISION RX, ADX, STF, TOLABS, TOLRA, TOLRI
DOUBLE PRECISION W, XP
DOUBLE PRECISION FUNCTION
DIMENSION X(NDIM)
DIMENSION W(10). IP(100)
DATA RXII. D-4I
DATA DMINI. D-351
DATA DXMAX/1.03/
DATA HR/1.0-1/
DATA HMIN/1.0-30/
DATA STF/1.00.00/
DATA RX/2.00/
DATA TOLR/1.0-5/
DATA TOLRA/1.0-71/
DATA TOLABS/0.00/

IEC = 0

10 CONTINUE
20 CONTINUE
C TAKE A RANDOM DIRECTION FOR THE DIRECTIONAL DERIVATIVE
CALL UNIJRV(XDIP,HF)
C COMPUTE FORWARD-DIFFERENCE DERIVATIVE
CALL DERFOR(NDIM, X, FV, DX, W, DDX)
C TRY THE FIRST HALF-STEP
DO 30 IC = 1, NDIM
      XP(1C) = X(1C) + W(1C)*DFDX + DBLE(FLOAT(NDIM))
1 CONTINUE
5 CONTINUE
HS = H
F = FUNCTION(NDIM, XP)
FS = FV + DABS(DFBX)
UPDATE STEPLength H AND ACCEPT OR REJECT THE HALF-STEP
CALL NEWM(XTIM, FS, F, H, IEC, IEC)
IF (IEC .LE. 0) GO TO 50
IF = IF - 1
IEC = IEC - 1
HS = HS
DFBX = DFBX
C COMPUTE CENTRAL-DIFFERENCES DERIVATIVE
SUBROUTINE SSTEP(KTIM, NYDIM, X, H, EPS, DX, IE, F)

C THE BASIC TIME-INTEGRATION STEP FOR A GIVEN TRAJECTORY IS PERFORMED

CALL THE SUBROUTINE STEP WHICH

CALLS THE FUNCTION FUNCTO TO COMPUTE THE VALUE OF F

CALL THE SUBROUTINE UNITRV TO COMPUTE THE RANDOM DIRECTION

ALONG WHICH THE DIRECTIONAL DERIVATIVE GAMAN IS TO BE
DO 40 IDD = 1,NTRAJ
1 1051. IF(ISVT(1D,1T) .EQ. ISVT(1D,1T)) NCV = NCV+1
2 1052. 40 CONTINUE
1 1053. 50 CONTINUE
1 1054. DO 80 IT = 2,NTRAJ
1 1055. 111 = IT-1
1 1056. NCV = 0
1 1057. DO 70 ID = 1,NTRAJ
1 1058. IF(ISVT(ID,1T) .EQ. ISVT(ID,1T)) NCN = NCN+1
2 1059. 70 CONTINUE
1 1060. IF(NCN .EQ. NCV) GO TO 110
1 1061. DO 60 ID = 1,NTRAJ
1 1062. IF(ISVT(ID,1T) .EQ. ISVT(ID,1T)) GO TO 100
1 1063. 60 CONTINUE
1 1064. DO 90 ID = 1,NTRAJ
1 1065. GO TO 100
1 1066. 90 CONTINUE
1 1067. 100 CONTINUE
1 1068. 100 CONTINUE
1 1069. 100 CONTINUE
1 1070. 100 CONTINUE
1 1071. DO 150 IT = 2,NTRAJ
1 1072. 150 CONTINUE
1 1073. VMVT(ID,1T) = VMVT(ID,1T) + VMVT(ID,1T)
1 1074. 150 CONTINUE
1 1075. DO 140 ID = 1,NTRAJ
1 1076. VMVT(ID,1T) = VMVT(ID,1T)
1 1077. 140 CONTINUE
1 1078. DO 130 IT = 1,NTRAJ
1 1079. 130 CONTINUE
1 1080. DO 120 IT = IT-1
1 1081. 120 CONTINUE
1 1082. VMVT(ID,1T) = VMVT(ID,1T)
1 1083. VMVT(ID,1T) = VMVT(ID,1T)
1 1084. 150 CONTINUE
1 1085. 160 CONTINUE
1 1086. RETURN
1 1087. END

SUBROUTINE STEP(ID)
1 1089. C STEP PERFORMS A SINGLE TIME-INTEGRATION STEP FOR EACH ONE OF THE
2 1090. C SIMULTANEOUS TRAJECTORIES BY REPEATEDLY CALLING THE SUBROUTINE STEP
1 1091. C DOUBLE PRECISION F
1 1092. C DOUBLE PRECISION X,H,DX,VMVT,FPS,VMCQ,VCOR
1 1093. C DOUBLE PRECISION XMIN,XMAX,XOPT,XOPT
1 1094. C DOUBLE PRECISION XID,MID,EPSID,DXID
1 1095. C DIMENSION XID(100)
1 1096. C COMMON /DINCOM/ X(100,20),H(20),DX(20),VMVT(20,19),EPS(20),
INTEGER FUNCTION IPRECE(ID1, ID2)

IPRECE determines the precedence relation between two trajectories based on their current value of EPS.

DOUBLE PRECISION X,H,DX,VMVT,EPS,VMCOR,VCOR
DOUBLE PRECISION XMVI,XRMAI,XOPT,FOPT
COMMON /DINCOM/ X(100,20),H(20),DX(20),VMVT(20,19),EPS(20),
   XMCOR(20),VOCOR(20),XMVI(100),XRIX(100),XOPT(100),FOPT, 
   J(20),ISVT(20,19),KGEN,KTIM,NDIM,NTRAJ,NTRAJR,
   ISG6BR,INKPBR,KPBRO,NCF,IFEP,INHP

IPRECE = 0
IF (KGEN*GT.INKPBR) GO TO 10
IF (EPS(ID2) .LT. EPS(ID1)) IPRECE = ID1
IF (EPS(ID2) .GT. EPS(ID1)) IPRECE = ID2
RETURN
10 CONTINUE

SUBROUTINE COMPAS
SUBPAS takes care of the storage of past history data, discarding all data not needed by the user of such data.

DOUBLE PRECISION X,H,DX,VMVT,EPS,VMCOR,VCOR
DOUBLE PRECISION XMVI,XRMAI,XOPT,FOPT
COMMON /DINCOM/ X(100,20),H(20),DX(20),VMVT(20,19),EPS(20),
   XMCOR(20),VOCOR(20),XMVI(100),XRMAI(100),XOPT(100),FOPT, 
   J(20),ISVT(20,19),KGEN,KTIM,NDIM,NTRAJ,NTRAJR,
   ISG6BR,INKPBR,KPBRO,NCF,IFEP,INHP

DO 30 IT = 2,NTRAJR
10 IF (IT1 = IT-1) GO TO 20
20 CONTINUE
104 CONTINUE
DO 50 ID = 1,NTRAJR
50 CONTINUE
DO 50 ID = 1,NTRAJR
50 CONTINUE
CONTINUE  

RETURN THE INDICES OF THE SEGMENTS WHICH IN THE ORDERING

C OCCUPY THE FIRST, THE LAST, AND A SUITABLE MEDIUM LEVEL POSITION

C  

IP = IORD(I)  
IU = IORD(NTRAJ)  
IM = IORD(ISEGBR)  

RETURN  
END

INTEGER FUNCTION IRREC(I011420)

IPREC DETERMINES THE PRECEDENCE RELATION BETWEEN TWO TRAJECTORIES

BASED ON THE PAST HISTORY DATA

DOUBLE PRECISION VM1, VM2  
DOUBLE PRECISION XM, DX, VMVT, EPS, VMG, G  
DOUBLE PRECISION XMIN, XMAX, XOPT, FOPT  
COMMON /INCOME/ X(100, 20), M(20), DX(20), VMVT(20, 100), EPS(20),  
VMG(20), VXOR(20), XMIN(100), XMAX(100), XOPT(100), FOPT;

IE(20), ISVT(20, 10), KS1E, KS2E, KS3E, KS4E, KS5E, INTP, NTRAJ, NTRAJR;

ISEGBR, INKBRA, IXBR, NCF, IFEP, INHP

DO 10 I = 1, NTRAJR  
II = IVTRAJR+II  
IF(ISVT(I01, II).EQ.ISVT(ID2, II)) GO TO 20  
VM1 = DMIN(VM1, VMVT(I01, II))  
VM2 = DMIN(VM2, VMVT(ID2, II))  

CONTINUE

IPREC = 0  
IF(VM2 .LT. VM1)IPREC = 102  
IF(VM2 .GT. VM1)IPREC = 101  

RETURN  
END
SUBROUTINE ORDER(IR,IM,1U)

ORDER COMPARES THE TRAJECTORIES FROM THE POINT OF VIEW OF PAST HISTORY (BY CALLING THE FUNCTION IPREC) AND FROM THE POINT OF VIEW OF THEIR CURRENT VALUE OF EPS (BY CALLING THE FUNCTION IPREC) AND PROVIDES THE TRAJECTORY ORDERING NEEDED FOR SELECTING THE TRAJECTORY WHICH IS TO BE BRANCHED.

DOUBLE PRECISION X,R,DX,VMX,EP0,VMX0,VMX0R
COMMON /DM70/ X(100,20),H(20),DX(20),VM,EP0(20),
VMX0(20),VMX0R(20),XMIN(100),XMAX(100),XOPT(100),FOPT,
1 IE(20),JSV1(20),KGEN,KTIN,NDIM,NTRA,J,NTRA0R,
15, K, Km,NDIM,NTRA1,INDIM,NTRA0R
1 COMMON /DIREM/ IORD(10)
DIMENSION IORD(10)
DATA IR=0/
IR = 0

C ASSIGN INITIAL ORDERING
C 10 CONTINUE
DO 20 I = 1,NTRA1
1 IORD(I) = I
20 CONTINUE

C SORT TRAJECTORIES ...

C 30 CONTINUE
IR = IR+1
C DO 60 I = 1,NTRA0R
1 I = I + 1
60 CONTINUE
K1 = IORD(I)
K2 = IORD(U)

C ... ACCORDING TO PAST HISTORY ...
IF(IR.NE.2)KP = IPREC(K1,K2)
IF(KP.EQ.0.AND.IR.EQ.1)GO TO 10

C ... OR ACCORDING TO NOISE LEVEL
IF(IR.EQ.2)KP = IPREC(K1,K2)
IF(KP.EQ.0)GO TO 40
KM = K1*K2-KP
100 K(I) = KP
1000(I) = K
855. C UPDATE PAST HISTORY DATA
856. C
857. C
858. DO 10 ID = 1, NTRAJ
859. NVT(I, ID, NTRAJ) = VMCOR(ID)
860. NVT(I, ID, NTRAJ) = ID
861. CONTINUE
862. C
863. C OBTAIN TRAJECTORY-SELECTION ORDERING
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, INAPBR) = EQ. EKBRO) IM = IP
870. C
871. C PERFORM BRANCHING
872. C
873. DO 20 IC = 1, NDIM
874. X(IC, IU) = X(IC, IM)
875. CONTINUE
876. H(IU) = H(IM)
877. IE(IU) = IE(IM)
878. DO 30 IT = 1, NTRAJ
879. ISVT(IT, IM, IU) = ISVT(IT, IM, IT)
880. VMVT(IT, IU, IU) = VMVT(IT, IM, IT)
881. CONTINUE
882. EPS(IU) = EPS(IM)
883. DX(IU) = DX(IM)
884. VMCOR(IU) = VMCOR(IM)
885. DO 40 ID = 1, NTRAJ
886. VMCOR(ID) = VMCOR(ID)
887. CONTINUE
888. C UPDATE PAST HISTORY DATA MATRICES
889. C
890. CALL COMPAS
891. C
892. C UPDATE SCALING DATA
893. C
894. CALL MOVSCA(IU, IM)
895. C
896. C UPDATE NOISE COEFFICIENT VALUES
897. C
898. IF (EPS(IU) > LE-0.00) GO TO 50
899. IF (IFEP = EQ. 2)
900. IF (IFEP = EQ. 2)
901. EPS(IU) = EPS(IU) + FACG**((CHAO5-1)-EFAC)
902. IF (IFEP = EQ. 1)
903. EPS(IU) = FACG**((OMIN1*DELPMX**))
904. EPS(IU) = ELSE(EPS(IU)**(CHAO5-1)-EFAC) + DLFACLO)
905. EPS(IU) = DMIN1(EPS(IU), EPSMAX)
906. CONTINUE
907. C UPDATE MAGNITUDE VOF SPATIAL DISCRETIZATION INCREMENT
908. C
909. DX(IU) = DX(IU) + FACG**CHAO5-1)
910. RETURN
911. C
SUBROUTINE BRASI

C BRASI PERFORMS THE SELECTION PROCESS FOR THE TRAJECTORIES
C PRASI - UPDATES THE DATA ABOUT THE PAST TRAJECTORIES
C ASKS FOR THE TRAJECTORY-SELECTION ORDERING BY CALLING THE
C SUBROUTINE ORDER
C - DISCARDS ONE OF THE TRAJECTORIES
C - PERFORMS BRANCHING ON ONE OF THE REMAINING TRAJECTORIES
C - MOVES THE DATA OF THE UNPERTURBED CONTINUATION
C TO THE POSITION OF THE PERTURBED CONTINUATION
C CALLS THE SUBROUTINE COMPAS TO EXAMINE DATA ABOUT PAST
C HISTORY OF THE TRAJECTORIES AND DISCARD IRRELEVANT DATA

DOUBLE PRECISION DRA, DLEPMX, DLFACL
DOUBLE PRECISION FRA, EPSMAX, FACF
DOUBLE PRECISION CHAOS
DOUBLE PRECISION X, H, DX, WMT, EPS, VMCOA, VCDR
DOUBLE PRECISION XRMIN, XRMAX, XOFT, FOPT
COMMON / DINO1 / XI(100), D(20), DX(20), WMT(20, 19), EPS(20),
1 VMCOA(20), VCDR(20), XRMIN(100), XRMAX(100), XOFT(100), FOPT,
2 IE(20), JSVT(20, 19), KGEN, KTSP, NDT, NTRAJ, NTRAJK,
3 ISLBR, INKPB, NAPR, NCF, IFEP, INHP

DATA FACF/10.00/
DATA FRA/.50/
DATA DFAC/1.03/
DATA EPSMAX/1.015/
DATA DLEPMX/.3010299956398119400/
DATA DLFACL/7.3010299956398119400/
CALL PERIOD

EXTRACT AND RESCALE SOME FINAL VALUES

FM = VCOR(1)
FMAX = VCOR(1)
IFM = 1
DO 30 ID = 2, NTRAJ
FMAX = DMAX1(FMAX, VCOR(ID))
IF(VCOR(ID) .GE. 6M) GO TO 20
FM = VCOR(ID)
IFM = ID
20 CONTINUE
30 CONTINUE

FM = FM
NCEF = NCF
CALL SEGSECA(IFM)
CALL VARPSCA(NX, XMN)

RETURN
END

SUBROUTINE PERIOD

PERIOD IS CALLED BY SUBROUTINE GENEVA TO PERFORM THE GENERATION
OF THE TRAJECTORY SEGMENTS.
PERIOD - COMPUTES THE DURATION OF THE OBSERVATION PERIOD, I.E. THE
NUMBER NMP OF ACCEPTED INTEGRATION STEPS IN A PERIOD
- COMPUTES ALL THE SEGMENT STEPS BY REPEATEDLY CALLING THE
SUBROUTINE STEP
- PERFORMS THE SEGMENT SELECTION BY CALLING THE SUBROUTINE
BRASI

DOUBLE PRECISION X, DX, VMN, EPS, VMCOR, VCOR

COMMON /DINCOM/ X(100), DX(20), VMN(20), EPS(20)
1 VMCOR(20), VMN(100), VMN(100), EPS(20), FOPT,
2 IE(20), ISV(20), IGEN, KTIN, NDIM, NTRAJ, NTRAJR,
3 ISNBR, INPBR, KPBRO, NCF, IEP, INHP

DETERMINE DURATION OF OBSERVATION PERIOD
(NUMBER OF TIME INTEGRATION STEPS)

KGEN = KGEN+1
NKGEN = 1
IF(INHP .EQ. 1) NKGEN =
SUBROUTINE GENEVA(NX,XMIN,FMIN,FMAX,NCEF)
C GENEVA PERFORMS THE GENERATION AND THE FINAL PROCESSING AND
C EVALUATION OF THE SET OF TRAJECTORY SEGMENTS CORRESPONDING TO
C THE CURRENT OBSERVATION PERIOD.
C GENEVA UPDATES THE SCALING ARRAYS DIST AND BIAS BY CALLING
C THE SUBROUTINES SE6SCA AND UPDSCA
C GENEVA GENERATES THE TRAJECTORY SEGMENTS BY CALLING THE SUB-
C ROUTINE PERIOD
C DETERMINES SOME END-OF-SEGMENT RESULTS (FPFMIN, FPFMAX,
C XPFMIN) USING THE RESCALING CAPABILITIES OF THE SUB-
C ROUTINES SE6SCA AND VARSCL.

DOUBLE PRECISION XMIN,FMIN,FMAX
DOUBLE PRECISION XD,XH,DX,WVVT,EPS,VMCOR,VCOR
DOUBLE PRECISION XMNM1,FRMAX,XOPT,FOPT
COMMON /DIMCOM/ X(100,20),H(20),DX(20),WVVT(20,19),EPS(20),
VMCOR(20),VCOR(20),XMIN(100),XMAX(100),XOPT(100),FOPT,
1E(20),IVST(20,19),XGEN,RTIM,NDIM,NTRAJ,NTRAJR,
1SEGGR,INKBPR,KPRRO,MCF,IFEP,INHP
DIMENSION XMIN(NX)

C UPDATE SCALING DATA
DO 10 ID = 1,NTRAJ
  CALL SE6SCA(ID)
  CALL UPDSCA(NX,X(1,ID))
CONTINUE
C GENERATE THE SIMULTANEOUS TRAJECTORY SEGMENTS
SUBROUTINE TRIAL(N,NMIN,NMAX,KPASCA,
   1 TOLREL,TOLABS,PRINT,XMIN,FMIN,
   2 FMAX,NFEW,NR,IGSTOR)

C THE SUBROUTINE TRIAL GENERATES A TRIAL, I.E., A SET OF COMPLETE
C SIMULTANEOUS TRAJECTORIES BY REPEATEDLY PERFORMING
C A CALL TO THE SUBROUTINE GENEVA WHICH GENERATES THE SET OF
C SIMULTANEOUS TRAJECTORY SEGMENTS CORRESPONDING TO THE CURRENT
C OBSERVATION PERIOD, AND PERFORMS THE TRAJECTORY SELECTION
C (POSSIBLE) CALL TO THE SUBROUTINE PTSEG WHICH PERFORMS
C END-OF-SEGMENT OUTPUT
C A CHECK OF THE TRIAL STOPPING CRITERIA (WITH THE AID OF THE
C FUNCTION TOLCM)
C A DECISION ABOUT ACTIVATING OR DEACTIVATING THE SCALING OF
C THE VARIABLES (ACTIONS PERFORMED BY CALLING THE SUBROUTINES
C ACTSCA AND NOSCA).
C DOUBLE PRECISION TOLREL,TOLABS,PRINT,XMIN,FMIN,FMAX
C DIMENSION XMIN(N)
C DATA IRNF/71/
C DO 20 IR = 1,NMAX
C ACTIVATE SCALING
C IF(IR.GE.KPASCA.AND.IR.GT.N*IRNF)CALL ACTSCA
C NR = IR
C GENERATE AND EVALUATE THE SIMULTANEOUS TRAJECTORY SEGMENTS
C PERIOD
C CALL GENEVA(N,XMIN,FMIN,FMAX,NFEV)
SUBROUTINE REINIT(nx,x0,eps0,irand,f,ife)

C REINIT PERFORMS THE INITIALIZATION OF ALL TRIALS FOLLOWING THE
C FIRST TRIAL, AND PART OF THE INITIALIZATION OF THE FIRST TRIAL.

C DOUBLE PRECISION x0,eps0,f
C DOUBLE PRECISION epsv,g
C DOUBLE PRECISION chaos
C DOUBLE PRECISION x,n,dx,vmv,eps,vncor,vcor
C DOUBLE PRECISION xmin,xmax,xopt,fopt
C COMMON /RINCOM/ x(1c0),20),n(20),dx(20),vmv(20),eps(20),
C 1 vncor(20),vcor(20),xmin(1c0),xmax(1c0),xopt(100),fopt(100),
C 2 ife(20),jsb(20),xkem,x0m,n,m,ntraj,ntrajr,
C 3 ise6br,inkpbr,inep,incf,ife
C DIMENSION x0(nx)
C DATA EPSV/1.00/
C C INITIALIZE RANDOM NOISE GENERATOR
C C G = CHAOS(IRAND)
C C IFEP = IFE
SUBROUTINE INIT(nx,x0,h0,eps0,dx0,irand,f0,n4,n2,n3,n4
1 ,inn,irf,xri,xra,iit)

C INIT PERFORMS THE INITIALIZATION OF THE FIRST TRIAL.
C THE PART OF THE INITIALIZATION WHICH IS COMMON ALSO TO THE TRIALS
C FOLLOWING THE FIRST ONE IS PERFORMED BY CALLING THE SUBROUTINE
REINIT.

C DOUBLE PRECISION XD,H0,DX,VMV,T,EPSP,VMCM,VCOR
C DOUBLE PRECISION XMIN,XMAX,XOPT,FOPT
C COMMON /DINCOM/ X(100,20),H(20),DX(20),VMV(20,19),EPS(20),
1 VMCM(19),VCOR(20),XMIN(100),XMAX(100),XOPT(100),FOPT,
2 KE(20),ISV(20,19),KGEN,KTIN,NDIN,NTRAJ,NTRAJR,
3 IS6BR,INKPBR,KPBR0,NCF,IEP,INHP
C DIMENSION XO(NX),XRIO(NX),XRA(NX)
C DATA KAMAX/100/
C DATA NTRAJM/20/
C DATA NTRAJ0/7/
C DATA INKPBR0/10/
C DATA KPBR00/3/
C CHECK PARAMETER VALUES
C
C IT = 0
C IF (NX.GT.NMAX .OR. NX.LT.1 .OR. H0.LE.0.DO .OR. EPS0.LE.0.DO
C 1 .OR. DX0.LE.0.DO) IT = -99
C IF (IT.EQ.-99) RETURN
C
C INITIALIZE SOME VARIABLES
C
C INHP = INH
C DO 10 IX = 1,NX
1 XMIN(IX) = XRI(IX)
1 XMAX(IX) = XRA(IX)
1 CONTINUE
C CALL NOSCA
C NTRAJ = N1
C IF(NTRAJ.EQ.0)NTRAJ = NTRAJ0
C NTRAJ = MIN(NTRAJ,NTRAJM)
C NTRAJ = MAX(NTRAJ,3)
C N1 = NTRAJ
C IS6BR = N2
SUBROUTINE NEWH(KV,FV,FH,IE,IEC)

NEWH IS CALLED BY THE SUBROUTINE STEP TO DECIDE WHETHER TO ACCEPT
OR REJECT THE FIRST HALF-STEP, AND TO PROVIDE AN UPDATED VALUE FOR H

DOUBLE PRECISION FV,FH
DOUBLE PRECISION FA,FMAX,MIN,H
DIMENSION FR(3),FA(4)
DATA FR/1.05D0,2.0D0,10.0D0/
DATA FA/1.0D0,2.0D0,2.0D0,2.0D0/
DATA MIN/1.0D-30/
DATA MMAX/1.0D25/
DATA IECMAX/50/

IF(FW.LT.F)GO TO 20

STEP ACCEPTED, POSSIBLY INCREASE THE STEPLENGTH H
R = FA(1)
IF(I-E+2.LT.K)R = FA(2)
IF(I-E+3.LT.K)R = FA(3)
IF(I-E.LT.0.AND.K.GT.9)R = FA(4)
H = R*H
GO TO 30

STEP REJECTED, DECREASE H
IC = MIND(3,IEC)
R = FR(IC)
H = H/R
CONTINUE

RETURN

END

SUBROUTINE DERFOR(LDIM,X,F,DX,W,DFDX)

DERFOR COMPUTED THE FORWARD FINITE-DIFFERENCES DIRECTIONAL
DERIVATIVES (CALLING FUNCTION)
DOUBLE PRECISION X,F,DX,W,DFDX
DOUBLE PRECISION DX,F,DFX,DFMAX,FM,S,XX
DOUBLE PRECISION FUNCT
DIMENSION X(NDIM),W(NDIM)
DIMENSION XX(100)
DATA DXFF/1.0/D,FDF/1.0/
DATA DXMAX/1.06/
CONTINUE
S = 0.6
DO 30 IC = 1,NDIM
XX(IC) = X(IC)*W(IC)*DF
S = S*(XX(IC)-X(IC)**2)
CONTINUE
IF(S.GT.0.0)GO TO 40
DX = DX*DFFF
GO TO 20
CONTINUE
FM = FUNCT(NDIM,XX)
DFDX = (FM-F)/DX
IF(DFDX.GT.DXMAX)RETURN
IF(DABS(DFDX).GT.1.0)RETURN
IF(DFDX**2.0.GT.0.0)GO TO 50
DX = DX*DF
GO TO 10
CONTINUE
RETURN
END

SUBROUTINE DERCE(NDIM,X,F,DX,W,DFDX)
C
C DERCE COMPUTED THE CENTRAL FINITE-DIFFERENCES DIRECTIONAL
C DERIVATIVES (CALLING FUNCT)
C
DOUBLE PRECISION X,F,DX,W,DFDX
DOUBLE PRECISION FM,F,DF,
DOUBLE PRECISION FUNCT
DIMENSION X(NDIM),W(NDIM)
DIMENSION XX(100)
CONTINUE
IF (IC = 1,NDIM
XX(IC) = X(IC)-W(IC)*DF
CONTINUE
FR = FUNCT(NDIM,XX)
FM = F+DFDX+DF
DFDX = (FM-FL)/(2.0*DF)
RETURN
SUBROUTINE RCLOPT(N, X0, FO)
C RCLOPT RECALLS THE CURRENT BEST MINIMUM VALUE FOPT FOUND SO FAR
C FROM ALGORITHM START, AND THE POINT XOPT (OR POSSIBLY ONE OF THE
C POINTS) WHERE FOPT WAS OBTAINED
C
DOUBLE PRECISION X0, FO
DOUBLE PRECISION X, DX, VMVT, EPS, VMCOR, VCOR
DOUBLE PRECISION XMN, XM, XM, XOPT, FOPT
COMMON /DIMCOM/ X(OPT, 20), XM(20), DX(20), VMVT(20, 19), EPS(20),
VMCOR(20), VCOR(20), XMN(100), XM(100), XOPT(100), FOPT,
OPT(20), NSY(20, 19), KEN, KTIN, NTM, NTJ, NRJ, NRJ.
1369. C SEGEB, INKPBN, KPRD, NCF, IFEP, INHP
1370. DIMENSION XO(N)
1371. C
1372. DO 10 I = 1, N
1373. X0(I) = XOPT(I)
1374. 10 CONTINUE
1375. FO = FOPT
1376. C
1377. RETURN
1378. END

SUBROUTINE STOOPT(N, X0, FO)
C STOOPT STORES THE CURRENT BEST MINIMUM VALUE FOPT FOUND SO FAR
C FROM ALGORITHM START, AND THE POINT XOPT (OR POSSIBLY ONE OF THE
C POINTS) WHERE FOPT WAS OBTAINED
C
DOUBLE PRECISION X0, FO
DOUBLE PRECISION X, DX, VMVT, EPS, VMCOR, VCOR
DOUBLE PRECISION XMN, XM, XM, XOPT, FOPT
COMMON /DIMCOM/ X(OPT, 20), XM(20), DX(20), VMVT(20, 19), EPS(20),
VMCOR(20), VCOR(20), XMN(100), XM(100), XOPT(100), FOPT,
OPT(20), NSY(20, 19), KEN, KTIN, NTM, NTJ, NRJ, NRJ.
1391. C SEGEB, INKPBN, KPRD, NCF, IFEP, INHP
1392. DIMENSION XO(N)
1393. C
1394. DO 10 I = 1, N
1395. XOPT(I) = X0(I)
1396. 10 CONTINUE
DOUBLE PRECISION FUNCTION FUNCT(N,XX)

C FUNCTO IS CALLED WHENEVER THE VALUE OF THE FUNCTION F IS REQUIRED
C IN THE NUMERICAL INTEGRATION PROCESS.
C THE FUNCTION FUNCT
C - RESCALES THE VARIABLES BY CALLING THE SUBROUTINE VARSCA
C - CALLS THE SUBROUTINE RANGE TO TAKE CARE OF THE CASES WHERE THE
C CURRENT POINT X IS OUTSIDE A GIVEN ADMISSIBLE REGION
C - CALLS THE USER-SUPPLIED FUNCTION FUNCT TO ACTUALLY COMPUTE THE
C VALUE OF F
C - POSSIBLY CALLS THE SUBROUTINE STOOPT TO UPDATE THE CURRENT
C BEST MINIMUM FUNCTION VALUE FOPT AND THE CORRESPONDING
C MINIMIZER XOPT

DOUBLE PRECISION XX
DOUBLE PRECISION F,X,KS
DOUBLE PRECISION FUNCTION
DOUBLE PRECISION XH,DX,VMVT,EPS,VMCOR,VMC
DOUBLE PRECISION XRMIN,XRMAX,XOPT,FOPT
COMMON /DINCON/ XH(100,20),DX(20),VMVT(20,19),EPS(20),
VMCOR(20),VMC(20),XRMIN(100),XRMAX(100),XOPT(100),FOPT,
N(100),JS(20),IJS(20,19),KGEN,NOR,NTRAJ,NTRAJ,
DIMENSION XX(N)
DIMENSION XS(100)
DO 10 IX = 1,N
10 CONTINUE
C DESCALE X-VARIABLES
CALL VARSCA(N,XX)
C CONSTRAIN THE X-VARIABLES WITHIN BOUNDS
CALL RANGE(N,XX,XRMIN,XRMAX)
C COMPUTE THE FUNCTION VALUE...
F = FUNCT(N,XX)+R
C ... AND POSSIBLY UPDATE THE BEST CURRENT MINIMUM
IF (F.LT.FOPT) CALL STOOPT(N,XX,F)
FUNC = F
NCF = NCF+1
RETURN
SUBROUTINE RANGE (XS, XRMIN, XRMAX, R)
C RANGE IS CALLED BY THE FUNCTION FUNCTO TO TAKE CARE OF THE CASES
C WHERE THE CURRENT POINT X IS OUTSIDE A GIVEN ADMISSIBLE REGION
C
DOUBLE PRECISION XS, XRMIN, XRMAX, R
DOUBLE PRECISION A, B, C, D, DLRMAX, RMAX, RR, XC
DIMENSION XS(N), XRMIN(N), XRMAX(N)
DATA RMAX /1.035/
DATA DLRMAX/8.5904782547915990D0/

C
R = 0.00
DO 40 I = 1,N
   A = XRMAX(I)
   C = XRMIN(I)
   XC = XS(I)
   IF (XC.LE.A) GO TO 10
   B = A+C-A
   RR = RMAX
   IF (XC.LT.B) RR = DEXP((XC-A)*DLRMAX/(B-A))-1.0D0
   R = R+RR
   XS(I) = XRMAX(I)
   GO TO 20
C
10 CONTINUE
C
GO TO 40
C
CONTINUE
C
CONTINUE
C
CONTINUE
C
RETURN
END

INTEGER FUNCTION ITOLOC(FMAX, FMIN, TOLER, TOLABS)
C ITOLOC DETERMINES WHETHER TWO QUANTITIES ARE TO BE CONSIDERED NUMERICAL
C EQUALLY EQUAL WITH RESPECT TO AN ABSOLUTE (OR RELATIVE) DIFFERENCE
1487. C (HI)RION, WITHIN GIVEN TOLERANCES
1488. C
1489. C DOUBLE PRECISION FM,FMJN,TOLREL,TOLABS
1490. C 1STOP = 0
1491. C
1492. C CHECK RELATIVE DIFFERENCE AGAINST TOLREL
1493. IF (DABS(FM-FMIN).LE.TOLREL*(DABS(FMIN)+DABS(FM))/2.0)
1494. 1 1STOP=1STOP+1
1495. C
1496. C CHECK ABSOLUTE DIFFERENCE AGAINST TOLABS
1497. IF (FM-FMIN.LE.TOLABS)ISTOP = ISTOP+2
1498. C ITOLCH = ISTOP
1500. C
1501. C RETURN
1502. C END

1507. C
1508. C INISCA INITIALIZES THE COMMON AREA /SCALE/ FOR THE SCALING DATA
1509. C DOUBLE PRECISION DISTS,BIAS,GRA,GRA
1510. C COMMON /SCALE/ DIST(10,10,20),BIAS(10,20),GRA(10,10,20),
1511. C GRA(10,20),GRA(20),ISCA,10SCA,NX,NORD
1512. C DATA NMSCA/10/
1513. C 1514. LSCA = -1
1515. IF(N.GT.NMNSCA.OR.N.EQ.7)RETURN
1516. NX = N
1517. NORD = ND
1518. JOSCA = 1
1519. DO J = 1,NX
1521. K = (N+J-1)/2
1522. DIST((K+1)*T1*IX,1,1) = 0.00
1523. CONTINUE
1524. DIST((K+1)*T1*IX,1,1) = 1.00
1525. BIAS((K+1)*T1*IX,1) = 0.00
1526. GRA((K+1)*T1*IX,1) = 0.00
1527. CONTINUE
1528. NGRA((K+1)*T1*IX,1) = 0
1529. CONTINUE
1530. C
1531. C RETURN
1532. C END
SUBROUTINE NOSCA

1534. C NOSCA DEACTIVATES THE SCALING
1535. C
1536. C
1537. DOUBLE PRECISION DIST,BIAS,GRA,GRA
1538. COMMON /SCALE/ DIST(10,10,20),BIAS(10,20),GRA(10,10,20),
1539. 1 GRA(10,20),GRA(20),LSCA,IDSCA,NX,NORD
1540. C
1541. C
1542. C
1543. C
1544. RETURN
1545. END

SUBROUTINE SEGSCA(30)

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

SUBROUTINE VARSCA(8,1)

1557. C VARSCA COMPUTES THE RESCALED VARIABLE AX + B
1558. C
1559. C
1560. C
1561. DOUBLE PRECISION X
1562. DOUBLE PRECISION XB
1563. DOUBLE PRECISION DIST,BIAS,GRA,GRA
1564. COMMON /SCALE/ DIST(10,10,20),BIAS(10,20),GRA(10,10,20),
1565. 1 GRA(10,20),GRA(20),LSCA,IDSCA,NX,NORD
1566. DIMENSION X(N),XB(10)
1567. C
1568. IF(LSCA.LE.0)RETURN
1569. DO 1 J = 1,N
1570. XB(1) = BIAS(1,IDSCA)
1571. DO 2 J = 1,N
1572. XB(1) = XB(1)+DIST(1,J,IDSCA)*X(J)
1573. 1 CONTINUE
1574. 2 CONTINUE
1575. RETURN
1576. END
2 1572. 2 CONTINUE
1 1573. 1 CONTINUE
1 1574. DO 1 1 = 1, N
1 1575. X(I) = XB(I)
1 1576. 1 CONTINUE
1 1577. C
1 1578. RETURN
1 1579. END

1581. SUBROUTINE CUMSCA(M,W,DFDX)
1582. C CUMSCA STORES CUMULATED STATISTICAL DATA ON THE ILL-CONDITIONING OF
1583. F(X+B) W.R.T. X
1584. C DOUBLE PRECISION W,DFDX
1585. C DOUBLE PRECISION DIS1,BIAS,GRAGRA,GRA
1586. COMMON /SCALE/ DIS1(1),0.20),BIAS(10,20),GRAGRA(10,10,20)
1587. 1 LRA(10,20),NGRA(20),LSCA,IDSCA,NX,NORD
1588. 1 DIMENSION W(N)
1589. 1 DATA DFBXMA/1.0D1/
1590. C
1591. IF(LSCA.LE.0)RETURN
1592. IF(DAUS(DFDX).GT.DFBXMA)RETURN
1593. DO 1 1 = 1, M
1594. DO 2 J = 1, N
1595. GRA(I,J,IDSCA) = GRA(I,J,IDSCA)+DFDX*W(I)*DFDX*W(J)
1596. 2 CONTINUE
1597. 1 CONTINUE
1598. C
1599. C
1600. C
1601. C
1602. C
1603. C
1604. RETURN
1605. END

1606. SUBROUTINE ACTSCA
1607. C ACTSCA ACTIVATES THE RESCALING
1608. C DOUBLE PRECISION DIS1,BIAS,GRAGRA,GRA
1609. C COMMON /SCALE/ DIS1(10,20),BIAS(10,20),GRAGRA(10,10,20)
1610. 1 GRA(10,20),NGRA(20),LSCA,IDSCA,NX,NORD
SUBROUTINE MOVSCA(JU, IM)

IF(LSCA.EQ.0) LSCA = 1
RETURN
END

SUBROUTINE MOVSCA(JU, IM)

MOVSCE MOVES THE SCALING DATA OF THE UNPERTURBED CONTINUATION TO THE
POSITION OF THE PERTURBED CONTINUATION

DOUBLE PRECISION DIST, BIAS, GRA, GRAGA
COMMON /SCALE/ DIST(10,10,20), BIAS(10,20), GRA(10,10,20)
    1 GRA(10,20), IGRA(20), LS CA, IDS CA, NX, NORD

IF(LSCA.LT.0) RETURN
DO 1 J = 1, NX
    DO 2 I = 1, JU
    DIST(I,J, IU) = DIS H (I,J, IN)
    GRA(I, J, IU) = GRAGRA(I, J, IN)
2 CONTINUE
    BIAS(I, IU) = BIAS(I, IN)
    GRA(I, JU) = GRA(I, JU)
1 CONTINUE

 RETURN
END

SUBROUTINE UPDSCA(N, X)

UPDSCA UPDATES THE SCALING MATRIX A AND THE BIAS VECTOR B BY
CALLING EIGSCA AND VARSCA

DOUBLE PRECISION X
DOUBLE PRECISION AGRAB, AAL, ALPHA, AMCOR, BIAS
DOUBLE PRECISION CH, COR, DISTT, SN
DOUBLE PRECISION EIGSCA
DOUBLE PRECISION DIST, BIAS, GRA, GRAGA
COMMON /SCALE/ DIST(10,10,20), BIAS(10,20), GRA(10,10,20)
    1 GRA(10,20), IGRA(20), LS CA, IDS CA, NX, NORD
DIMENSION X(N)
DIMENSION DISTT(10,10), BIAS(10), COR(10,10)
DATA ALPHA /0.3001/
1655. C
1656. IF(LSCA.LE.0)RETURN
1657. ID = LSCA
1658. IF(NGRA(ID).LT.2*NX*NX)GO TO 2
1659. AGRA1 = 1./Dble(NUAT(NGRA(ID)))
1660. AMCOR = 0.,00
1661. DO 1 J = 1,NX
1662. DO 4 I = 1,NX
1663. COR(I,J) = AGRA1*GRA(I,J,1D)-AGRA1*GRA(I,1D)*AGRA1*GRA(J,1D)
1664. AMCOR = COR(I,J)/AMCOR
1665. 4 CONTINUE
1666. CONTINUE
1667. IF(AMCOR.LE.0.,DO 11 J = 1,NX
1668. DO 11 J = 1,NX
1669. COR(I,J) = COR(I,J)/AMCOR
1670. 11 CONTINUE
1671. CONTINUE
1672. COR(I,J) = 1.0*Dble(ID)
1673. DO 12 I = 1,NX
1674. COR(I,J) = 1.0*Dble(ID)
1675. DO 12 J = 1,NX
1676. 12 CONTINUE
1677. CALL VARSCA(NX,BIAST)
1678. SN = 0.0
1679. DO 6 J = 1,NX
1680. DO 6 I = 1,NX
1681. DIST(I,J) = 0.0
1682. DO 8 K = 1,NX
1683. DIST(I,J) = DIST(I,J)-K*DST(I,K,1D)*COR(K,J)
1684. 8 CONTINUE
1685. SN = SN*DIST(I,J)**2
1686. 6 CONTINUE
1687. CONTINUE
1688. SN = 1./Dble(DSQRT(SN/Dble(NUAT(NX))))
1689. DO 9 I = 1,NX
1690. BIAS(I,1D) = BIAST(I)
1691. DO 10 J = 1,NX
1692. LIST(I,J,1D) = SN*DIST(I,J)
1693. BIAS(I,J) = BIAS(I,1D)-DIST(I,J,1D)*LIST(I,J)
1694. 10 CONTINUE
1695. CONTINUE
1696. GRA(I,J,1D) = 0.,00
1697. 9 CONTINUE
1698. GRA(I,1D) = 0.,00
1699. 0 CONTINUE
1700. NGRA(Id) = 0
1701. CONTINUE
1702. CONTINUE
1703. C
1704. RETURN
1705. END
DOUBIV PRECISION FUNCTION EIGSCA(COR)

EIGSCA COMPUTES THE LARGER GIGENVALUE OF A MAFIX USEA FOR RESCALING, STARTING FROM A RANDOMLY-CHOOSEN ESTIMATE (OBTAINED BY CALLING THE SUBROUTINE UNIFR) OF THE CORRESPONDING EIGENVECTOR.

DOUBIV PRECISION COR
DOUBIV PRECISION ALA1, ALA11, ALA10
DOUBIV PRECISION PREC, SW1, WW
DOUBIV PRECISION DIST, BIAS, GRA, GRA1

COMMON /SCALE/ DIST(10, 10, 20), BIAS(10, 20), GRA(10, 10, 20), GRA1(10, 10, 20),

1 GRA(10, 20)+GRA1(20)+LSCA, IDSCA, NX, NORD

DIMENSION COR(10, 10)
DIMENSION WI(10), WW(10)
DATA PREC /1. D-3/
DATA NRMIN /10/
DATA NRMAX /100/

CALL UNIFR(NX, W)
ALA1 = 0 . D0
DO 1 JR = 1, NRMAX

1 ALA10 = ALA1
SW = 0 . D0
DO 2 IX = 1, NX

2 WW(IX) = 0 . D0
DO 3 JX = 1, NX

3 WW(IX) = WW(IX)+COR(IX, JX)*W(JX)
CONTINUE

SW = SW+WW(IX)**2
CONTINUE

1 ALA1 = DSQRT(SW)
ALA11 = 1 . D0/ALA1

1 IF(IJX.EQ.NRMIN .AND. ALA1*PREC, GT, DABS(ALA1-ALA10)) GO TO 4

1 DO 5 IX = 1, NX

5 WW(IX) = WW(IX)*ALA11
CONTINUE

1 CONTINUE

4 CONTINUE

EIGSCA = ALA1
RETURN
END

DOUBIV PRECISION FUNCTION CHAOS(IN12)
CHAOS GENERATES A RANDOM SAMPLE FROM ONE OUT OF FOUR POSSIBLE PROBABILITY DISTRIBUTIONS USING RANDOM NUMBERS UNIFORMLY DISTRIBUTED IN (0, 1) GENERATED BY THE FUNCTION UNIFR.
DOUBLE PRECISION FUNCTION UNIFRD(INIZ)

C UNIFRD GENERATES THE RANDOM NUMBERS UNIFORMLY DISTRIBUTED IN (0,1)
C EXPLOITING THOSE GENERATED BY ALKUT WITH A FURTHER RANDOMIZATION

DOUBLE PRECISION UNIFRD,PAI,A,B

DATA PAI/3.14159265358979323846/ 

IF(INIZ.LE.0) GO TO 10 

C INITIALIZATION.

CHADOS = UNIFRD(INIZ)
RETURN 

10 CONTINUE 

A = UNIFRD(0)
B = UNIFRD(0)

C GAUSSIAN RANDOM NUMBER BY POLAR METHOD

CHADOS = SQRT(-2.D0*LOG(A))*DCOS(PAI*B)
RETURN 

20 CONTINUE 

C UNIFORM RANDOM NUMBER IN (0,1)

CHADOS = A

C CAUCHY RANDOM NUMBER BY INVERSE TRANSFORMATION

IF(INIZ.EQ.(-1)) CHADOS = DSIN(PAI*A)/DCOS(PAI*A)

C UNIFORM RANDOM NUMBER IN (-1,1)

IF(INIZ.EQ.(-2)) CHADOS = 2.D0*A-1.D0
RETURN 
END
1803. C IF THE INPUT PARAMETER INITZ IS NOT 0
1804. C THE RANDOM NUMBER GENERATOR IS INITIALIZED
1805. C
1806. C DOUBLE PRECISION A,B,C,X
1807. C DOUBLE PRECISION X0,P,P0,P1,P2,R1,R2
1808. C DOUBLE PRECISION FINV
1809. C
1810. C DIMENSION X(61)
1811. C
1812. C DATA NREM/61d.NREM/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 P0/3.00,P1,P2/1.00,3.00,R1,R2/0.250,0.750/
1817. C
1818. C IF(INIZ.NE.0.or.IREM.EQ.0) GO TO 10
1819. C
1820. C IPO = IREM
1821. C X0 = X(10)
1822. C
1823. C NONLINEARIZATION OF X0 TO AVOID LONG-DISTANCE LINEAR RELATIONSHIP
1824. C
1825. C IF(X0.GE.FINV)X0 = DMOD(X0/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(10)+1.0D0)
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 = P0-1.0D0/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) = P*(k2-R1)*(P-P1)/(P2-P1)
1848. C 20 CONTINUE
1849. C
1850. C IREM = 0
1851. C
1852. C DO 30 K = 1,NREM
1853. C
1854. C CALL ALKNUT(NREM,X,IREM)
1855. C
1856. C 30 CONTINUE
1857. C
1858. C UNIFRD = X(1)
1859. C

37
SUBROUTINE ALKNUIT(NREM, X, IREM)

C UPDATES THE COMPONENT IREM OF THE NREM-VECTOR X WITH A RANDOM NUMBER
C UNIFORMLY DISTRIBUTED IN (0,1) BY MEANS OF THE ALGORITHM
C OF MITCHELL-MOORE, MODIFIED AS SUGGESTED BY BRENT, QUOTED IN
C SECOND VOLUME, SEMINUMERICAL ALGORITHMS, ADDISON-WESLEY
C DOUBLE PRECISION X
C IF(IREM.NE.0) GO TO 10
C IREM = NREM
C 17 = NREM - N1
C 12 = NREM - N2
C 10 CONTINUE
C X(IREM) = (MOD(X(17)+X(12),1.0D0)
C IREM = 1+MOD(IREM,NREM)
C 17 = 1+MOD(17,NREM)
C 12 = 1+MOD(12,NREM)

C RETURN
C END

SUBROUTINE GAUSVR(N,M)

C GENERATES A RANDOM VECTOR SAMPLE FROM AN M-DIMENSIONAL
C NORMAL DISTRIBUTION
C DOUBLE PRECISION X,Y,Z
C DOUBLE PRECISION CHAOS

RETURN
END
3.- APPLICAZIONE DELL'ALGORITMO ALL'ANALISI CONFORMAZIONALE

L'algoritmo descritto nel paragrafo 2 è stato concepito un uso del tutto generale.

Per lo studio delle posizioni di equilibrio delle molecule nel caso di geometria di valenza rigida, la funzione da minimizzare è l'energia conformazionale

\[ E(\phi) = E(\phi_1, ..., \phi_m) \]

in cui gli argomenti \( \phi_1, ..., \phi_m \) sono gli angoli di torsione liberi.

Il calcolo effettivo di \( E(\phi) \) viene effettuato come segue.

Dati gli angoli \( \phi_1, ..., \phi_m \) si calcola direttamente il contributo torsionale alla energia \( E(\phi) \), 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 \( E(\phi) \) 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 \( \epsilon, \zeta, \alpha, \beta, \gamma \), 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 (numericalmente) 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, 
\[ t_k = h_0 + h_1 + h_2 + \ldots + h_{k-1}, \]
\( r_k \) e \( u_k \) 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 fi-
nite della derivata direzionale nella direzione \( r_k \).

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 mante-
nuto costante, mentre \( h_k \) e il passo \( \Delta x_k \) usato per calcola-
re \( \tilde{\eta}_k \) sono aggiustati automaticamente per ciascuna traiettoria.

Al termine di ogni periodo di osservazione le traietto-
rrie 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 "di-
ramazione", e cioè a una seconda continuazione della stessa 
traiettoria, che differisce dalla prima solo per i valori 
iniziali di \( \epsilon \) e \( \Delta x \), ma che si considera avere la stessa 
"storia passata" della prima.

Il numero totale di traiettorie simultanee rimane per-
ciò invariato, e la seconda continuazione prende - dal punto 
di vista del programma di calcolo - il posto della traietto-
rria scartata.

L'insieme delle traiettorie simultanee è considerato 
come una singola "prova", e l'algoritmo completo è un insie-
me di prove ripetute.
L'equazione (1) si può considerare come caso limite del l'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 \(x(t)\) soluzione dell'equazione (1) con \(\epsilon\) costante, a partire da un punto iniziale \(x_0\): si ha infatti che sono ipotesi molto poco restrittive sulla funzione \(f\) - la densità di probabilità di \(x(t)\) all'istante \(t\) tende, per \(t \to \infty\), a una densità di probabilità limite

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

indipendentemente dal punto iniziale \(x_0\) (A è una costante di normalizzazione), che risulta tanto più concentrata intorno ai minimi globali di \(f\) quanto minore è \(\epsilon\), fino a divenire, nel limite per \(\epsilon \to 0\), una somma pesata di delta di Dirac centrata 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 \(\epsilon \to 0\), 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 \(x_0\) e la discretizzazione usata per la (1) ha la forma

\[
x_{k+1} = x_k - h_k \eta \frac{\Delta x_k}{\Delta t_k} + \epsilon(t_k)\sqrt{h_k} u_k \quad k = 0,1,2,\ldots
\]
2.- DESCRIZIONE DELL'ALGORITMO PER LA RICERCA DEL MINIMO GLOBALE

L'algoritmo di minimizzazione globale adottato, che si basa sul metodo proposto da F. Aluffi-Pentini, V. Parisi, e F. Zirilli in [1], è descritto in dettaglio in [2], mentre la sua traduzione in un insieme di sottoprogrammi FORTRAN è in [3].

Il metodo ricerca un punto di minimo globale di una funzione \( f(x) = f(x_1, x_2, ..., x_n) \) di \( n \) variabili reali, seguendo le traiettorie generate da una opportuna discretizzazione numerica dell'equazione differenziale stocastica.

\[
\frac{dx}{dt} = -\nabla f(t) + \epsilon \, dw
\]

[a partire da una condizione iniziale \( x(0) = x_0 \)] essendo \( \nabla f \) il gradiente di \( f(x) \), \( w(t) \) un processo stocastico di Wiener standardizzato a \( n \) dimensioni, e \( \epsilon \) un coefficiente positivo ("coefficiente di rumore") che consideriamo variabile nel tempo.

L'equazione (1) - generalmente considerata con \( \epsilon \) 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 \( \epsilon \, dw \), 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.
2.

La possibilità 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 $E(x)$ che dovrebbero essere eseguiti richiede 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 $E(x)$ 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 $15^\circ$ per ogni legame ruotato: questo comporta il calcolo dell'energia in circa $8 \times 10^6$ 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 \( \mathbf{x} = (x_1, x_2, \ldots, x_{3N-6}) \) (dove \( N \) è il numero di atomi contenuti nella molecola), nei quali l'energia potenziale intramolecolare \( E(\mathbf{x}) \) 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 \( E(\mathbf{x}) \) nell'iperspazio conformazionale è di maggiore interesse in chimica o biologia il minimo o i minimi globali cioè i punti \( \mathbf{x}^* \) tali che per ogni \( \mathbf{x} \) sia:

\[
E(\mathbf{x}^*) \leq E(\mathbf{x})
\]

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
Istituto Guido Donegani s.p.a., Via G. Fauser, 4, 28100 Novara

Filippo Aluffi-Pentini
Dipartimento di Matematica, Università di Bari, 70125 Bari

Valerio Parisi
Dipartimento di Fisica, II Università di Roma (Tor Vergata), 00173 Roma

Francesco Zirilli
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).
C MAIN PROGRAM (SAMPLE VERSION)
C CALLS SIGMA VIA THE DRIVER SUBROUTINE SIGMA1
C
C DOUBLE PRECISION X0,XMIN,FMIN
C
DIMENSION X0(2),XMIN(2)
C
C TEST PROBLEM DATA
C
C PROBLEM DIMENSION
N = 2
C
C INITIAL POINT
X0(1) = 0.00
X0(2) = 0.00
C
C SET INPUT PARAMETERS
NSUC = 3
IPRINT = 0
C
CALL DRIVER SUBROUTINE SIGMA1
CALL SIGMA1(N,X0,NSUC,IPRINT,XMIN,FMIN,NFEV,IOUT)
C
STOP
END

DOUBLE PRECISION FUNCTION FUNCT(N,X)
C COMPUTES THE VALUE AT X OF THE SIX-HUMP CAMEL FUNCTION

C
DIMENSION XX,YY
XX = X(1)*X(1)
YY = X(2)*X(2)
FUNCT = (XX/3.0D0-2.0D0)*XX+4.0D0)*XX+X(1)*X(2)
* +4.0D0*(YY-1.0D0)*YY
RETURN
END
DOUBLE PRECISION FUNCTION FUNCT(N,X)
C Computes the function values of test-problem MPROB
C by calling the subroutine GLOMF.
C
DOUBLE PRECISION X
DOUBLE PRECISION F
DIMENSION X(N)

COMMON /IUN/ NPROB
CALL GLOMF(NPROB,N,X,F)
FUNCT = F
RETURN
END
C (ALGORITHM SIGMA)

C MAIN PROGRAM (TEST VERSION)

C (CALL SIGMA VIA THE DRIVER SIGMA1)

C DOUBLE PRECISION FMIN, X0, XMAXGL, XMIN, XMINGL

C COMMON AREA TO PASS TEST-PROBLEM NUMBER NPROB

C TO THE FUNCTION FUNCT WHICH WILL COMPUTE

C THE FUNCTION VALUES OF TEST-PROBLEM NPROB

C BY CALLING THE TEST-PROBLEM COLLECTION SUBROUTINE GLOMIP

C COMMON  /IUN/ NPROB

C XO INITIAL POINT

C XMIN FINAL ESTIMATE OF GLOBAL MINIMUM

C XMINGL XMAXGL MUST BE DIMENSIONED HERE IN ORDER TO CALL

C THE PRE-EXISTING SUBROUTINE GLOMIP.

C DIMENSION X0(100), XMIN(100), XMINGL(100), XMAXGL(100)

10 CONTINUE

C INPUT PROBLEM NUMBER

WRITE(6,20)

20 FORMAT(///4 HH INPUT PROBLEM NUMBER (1 TO 37, 0 = STOP))

READ(5,30)NPROB

29 FORMAT(///18 HH PROBLEM NUMBER = ,12/////)

30 CONTINUE

31 C TERMINATE OR CONTINUE

32 IF(NPROB.EQ.0) GO TO 50

33 C CALL GLOMIP TO GET PROBLEM DIMENSION N AND INITIAL POINT XO

34 C NOTE THAT GLOMIP RETURNS ALSO THE BOUNDARIES XMINGL, XMAXGL

35 C OF THE OBSERVATION REGION (NOT NEEDED HERE)

36 CALL GLOMIP(NPROB, N, XO, XMIN, XMINGL, XMAXGL)

38 C

39 C SET NSUC SO AS TO HAVE GOOD CHANCES, WITHOUT PROHIBITIVE

40 C COMPUTATIONAL EFFORT

NSUC = 5

44 C

47 C SET IPRINT SO AS TO HAVE A MODERATE OUTPUT

IPRINT = 0

49 C

52 C CALL DRIVER SUBROUTINE SIGMA1

CALL SIGMA1(N, XO, NSUC, IPRINT, XMIN, XMIN, NFE, IOUT)

56 C

GO TO THE NEXT PROBLEM

50 GO TO 10

55 C

57 C END OF TEST PROBLEMS

54 CONTINUE

58 WRITE(6,60)

60 FORMAT(///22 HH END OF TEST PROBLEMS /)

40
SUBROUTINE UNITRV(N,W)
  C
  C       GENERATES A RANDOM VECTOR UNIFORMLY DISTRIBUTED
  C       ON THE UNIT SPHERE.
  C
  C       DOUBLE PRECISION W,WW
  C
  DIMENSION W(N)
  CALL GAUSRV(N,W)
  WW = 0.D0
  DO 10 I = 1,N
    WW = WW+W(I)**2
  10 CONTINUE
  WW = 1.D0/DSQRT(WW)
  DO 20 I = 1,N
    W(I) = WW*W(I)
  20 CONTINUE
  RETURN
END
tenendo la geometria di valenza costante (con le lunghezze di legame e gli angoli di valenza risultanti dalla analisi cristallografica della citosina-3'-fosfato\(^\text{[6]}\)). Mediante un procedimento di best-fit con tali energie fu trovata una funzione di potenziale (abbreviata TCM dalle iniziali degli autori del rif.\(^\text{[5]}\)), 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:

\[
E(\phi) = \sum_{1 \neq j} \left( \frac{A_{ij}}{r_{ij}} + \frac{B_{ij}}{r_{ij}^6} \right) + \sum_{i=1}^{l} \frac{1}{2} K_i \left( 1 + \cos 3\phi \right),
\]

dove i \(\phi\) sono gli angoli di torsione \(\gamma, \beta, \alpha, \tau, \epsilon\) espressi in radianzi e dove la somma \(\sum\) è estesa alle coppie di atomi non legati e dove si è assunto nullo il contributo elettrostatico.

I valori numerici dei parametri \(A, B, E\) sono riportati nella Tabella 1 del rif.\(^\text{[7]}\). 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 \(\gamma = 176,2^\circ\), \(\beta = 180,0^\circ\), \(\alpha = 122,1^\circ\),...
$\beta = -96,7^\circ, \gamma = 55,6^\circ, E = -94,0 \text{ KJmol}^{-1}$), caratterizzato da contatti $H(C3')...O6 = 1,75 \ \text{\AA}, \ H(C2')...O3 = 1,82 \ \text{\AA}, \ H(C2')...O5' = 2,47 \ \text{\AA}$. Si osservi che un procedimento di ricerca "diretta" dei minimi di più bassa energia 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 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 posti, 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 perché 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


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 ommessi.