Topics in Global Optimization

Ada Montalvo
Northwestern University
6 PUBLICATIONS **706** CITATIONS
SEE PROFILE

Susana Gómez
Universidad Nacional Autónoma de México
113 PUBLICATIONS **1,766** CITATIONS
SEE PROFILE

TOPICS IN GLOBAL OPTIMIZATION

A. V. Levy, A. Montalvo,
S. Gomez and A. Calderon.

IIMAS - UNAM.
A.P. 20-726, Mexico 20, D.F.
Mexico City, Mexico.

ABSTRACT. A summary of the research done in global optimization at the Numerical Analysis Departament of IIMAS-UNAM is given. The concept of the Tunnelling Function and the key ideas of the Tunnelling Algorithm as applied to Unconstrained Global Optimization, Stabilization of Newton's Method and Constrained Global Optimization are presented. Numerical results for several examples are given, they have from one to ten variables and from three to several thousands of local minima, clearly illustrating the robustness of the Tunnelling Algorithm.

1. UNCONSTRAINED GLOBAL OPTIMIZATION. (Refs. 1,2).

1.1 Statement of the problem.

In this section we consider the problem of finding the global minimum of $f(x)$, where f is a scalar function with continuous first and second derivatives, x is an n-vector, with $A \leq x \leq B$, where A and B are prescribed n-vectors.

In this work we assume that the problem does have a solution and if we say that $f(x^*)$ is the global minimum, this means that a) $f(x^*)$ is a local minimum, satisfying the optimality conditions $f_x(x)=0$, $f_{xx}(x)$ p.d., and b) at x^* the function has its lowest possible value, satisfying the condition $f(x^*) < f(x)$,for $A \leq x \leq B$.

The particular cases, when there are several local minima at the same function level, or the lowest value of the function occurs at the boundary of the hypercube $A \leq x \leq B$, were considered in the original paper, (Ref.2) but for brevity they are omitted in this section. The type of functions $f(x)$ being considered, are illustrated in Figs. 1 and 2.

1.2 Description of the Tunnelling Algorithm.

Design Goal. Since the type of problems considered in global optimization have a large number of local minima, the design goal of this algorithm is to achieve a Generalized Descent Property, that is, find sequentially local minima of f(x) at x_i^* , i=1,2,...,G, such that

$$f(x_i^*) \geqslant f(x_{i+1}^*) \quad , \quad A \leqslant x_i^* \leqslant B \quad , \quad i=1,2,\ldots,G-1 \tag{1}$$

thus avoiding irrelevant local minima, approaching the global minimum in an orderly fashion.

Structure of the Tunnelling Algorithm. The Tunnelling Algorithm is composed of a sequence of cycles, each cycle consists of two phases, a minimization phase and a tunnelling phase.

The minimization phase is designed to decrease the value of the function. For each given nominal point x_i°, i=1,2,.. ,G, find a local minimum say at x_i^* , that is solve the problem

$$\min_x f(x) = f(x_i^*) \quad , \quad i=1,2,\ldots,G \tag{2}$$

Any minimization algorithm, with a local descent property on f(x) can be used in this phase.

The tunnelling phase is designed to obtain a good starting point for the next minimization phase. Starting from the nominal point x_i^* , i=1,2,..G-1 we find a zero of the "TunnellingFunction",T(x,Γ), that is

$$T(x_{i+1}^\circ,\Gamma) \leqslant 0 \quad , \quad i=1,2,\ldots G-1 \tag{3}$$

using any zero finding algorithm.

1.3 Derivation of the Tunnelling Function.

We define the Tunnelling Function as follows

$$T(x,\Gamma) = \frac{f(x) - f(x^*)}{\{(x-x^*)^T(x-x^*)\}^\lambda} \tag{4}$$

where Γ denotes the set of parameters $\Gamma=(x^*, f(x^*),\lambda)$. Why we arrive at this definition can be seen in the following derivation:

a)-Consider Fig. 3 .a. Let x^* be the relative minimum found in the last minimization phase. We would like to find a point x°, $x^\circ \neq x^*$, $A \leqslant x^\circ \leqslant B$, where

$$f(x^\circ) \leqslant f(x^*) = f^* \tag{5}$$

This point x°would be a very good nominal point for starting the next

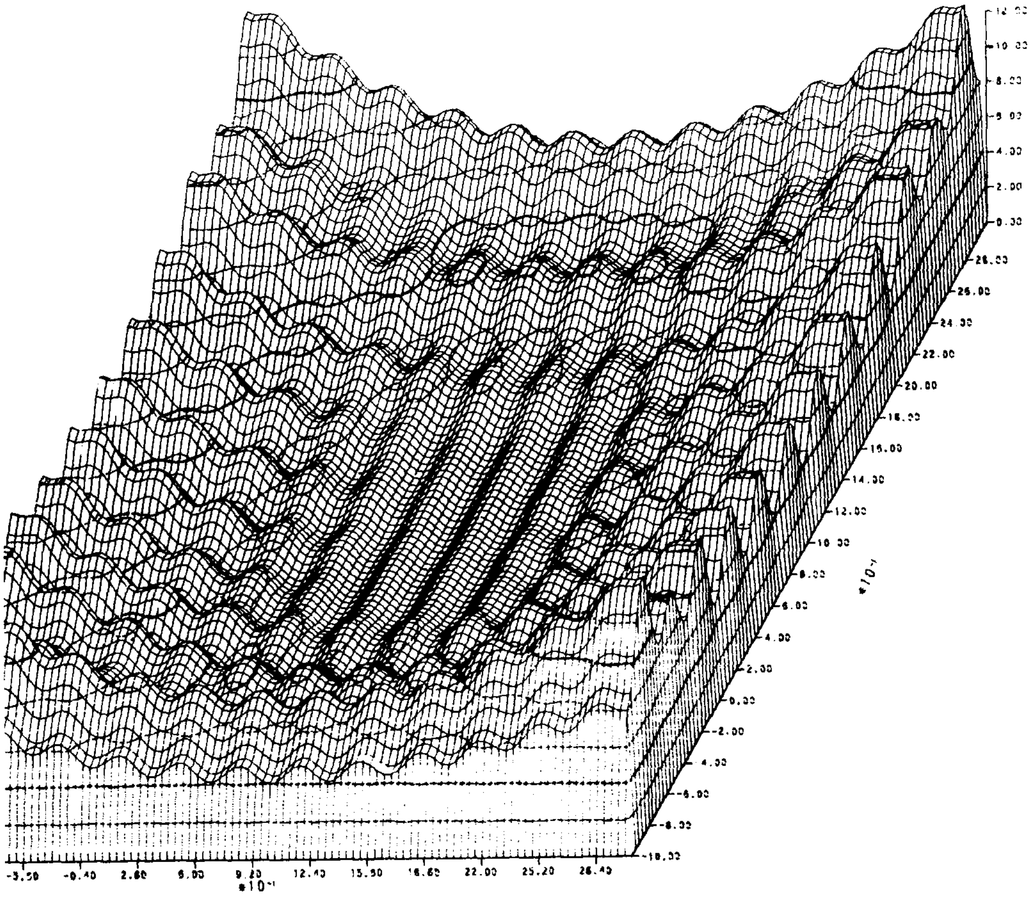

Figure 1. Typical function f(x) being minimized, with 900 local minima.This is a partial illustration of Example 13.

minimization phase,because,obviously,the next minimization phase will produce a local minimum with a lower function value than $f(x^*)$.
So the first thing to do is to store the value of $f(x^*)$ and shift the function, obtaining the expression

$$T(x,f^*) = f(x) - f(x^*) \qquad (6)$$

b)-Consider now Fig. 3.b. We have now $T(x^*,f^*) = 0$, $T(x°,f^*) = 0$, without having perturbed the position of x°. This being the case, we can find x°, if we look for a zero of $T(x,f^*) = 0$.However to avoid being attracted by the zero at x^*, we store the value of x^* and cancell this zero by introducing a pole at x^*; thus we obtain the expression

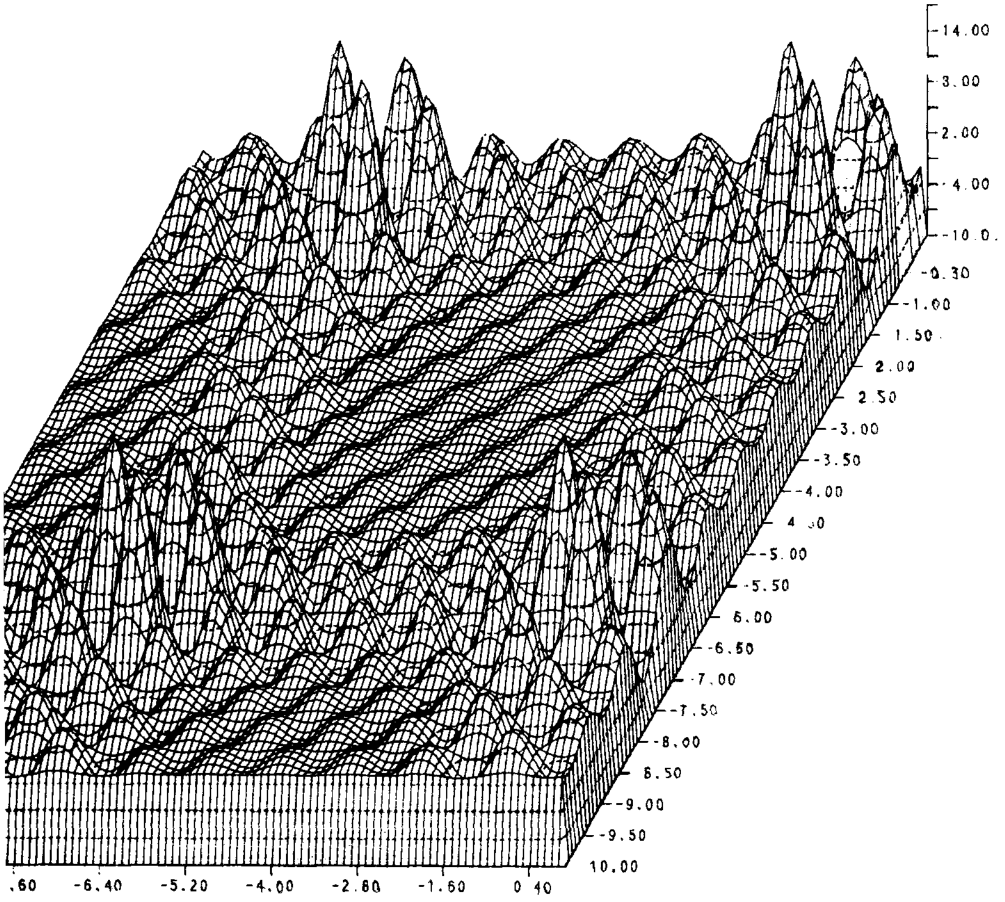

Figure 2. Typical function being minimized, with 760 local minima.This is a partial illustration of Example 3. taking -f(x)/100.

$$T(x,x^*,f^*,\lambda) = \frac{f(x) - f(x^*)}{\{(x-x^*)^T(x-x^*)\}^\lambda} \qquad (7)$$

where λ denotes the strength of the pole introduced at x^*. If we let Γ denote the parameters x^*,f^*,λ we obtain the definition of the Tunnelling Function given in Eq.(4).Fig. 3 .c illustrates the final effect of mapping f(x) into $T(x,\Gamma)$, once the the zero at x^* has been cancelled,

1.4 Computation of the pole strength.

The correct value of λ is computed automatically,starting with $\lambda = 1$ and

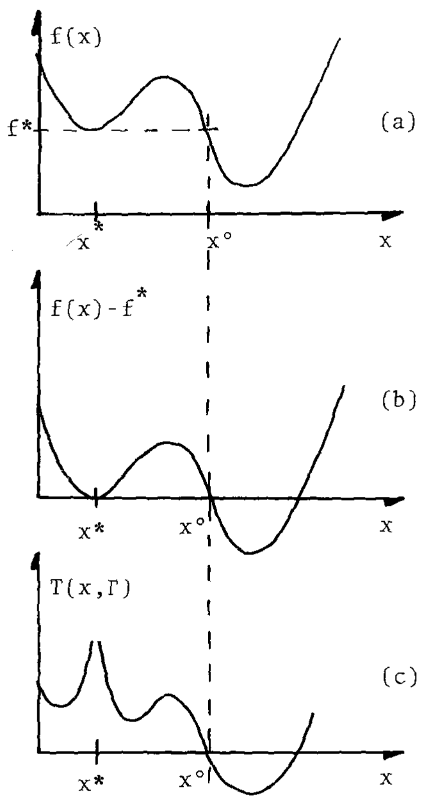

Fig. 3 . Derivation of the Tunneling Function.

the value of $T(x,\Gamma)$ at $x=\overset{*}{x}+\varepsilon$,where $\varepsilon<<1$ is used just to avoid a division by zero.If $T(x,\Gamma)>0$, λ has the correct value.If $T(x,\Gamma)=0$, λ is incremented,say by 0.1 until $T(x,\Gamma)>0$.

1.5 Generalized Descent Property.

Due to the definition of the minimization and tunneling phases, and their sequential use as shown in Fig.4.(a) the following relationship holds

$$f(x_i^\circ)\geqslant f(\overset{*}{x}_i)\geqslant f(x_{i+1}^\circ)\geqslant f(\overset{*}{x}_{i+1}) \quad ,i=1,2..G-1$$

and dropping the intermediate points x_i° ,the following Generalized Descent Property is obtained

$$f(\overset{*}{x}_i) \geqslant f(\overset{*}{x}_{i+1}) \quad , \quad i=1,2,\dots,G-1 \qquad (8)$$

Having satisfied our design goal for the Tunnelling Algorithm let's look at the numerical experiments.

1.6 Numerical Experiments.

Some of the numerical examples are listed in the appendix.The Tunnelling Algorithm was programmed in FORTRAN IV, in double precision.A CDC-6400 computer was used.
The Ordinary Gradient Method was used in the minimization phase.The stepsize was computed by a bisection process, starting from $\alpha=1$,enforcing a local descent property on $f(\alpha)< f(0)$ .Nonconvergence was assumed if more than 20 bisections were required.The minimization phase was terminated when $f_x^T(x)f_x(x)< 10^{-9}$.
In the tunnelling phase the Stabilized Newton's Method was employed,(see section 2. and Refs. 3, 4).The stopping condition for the tunnelling phase was taken as $T(x,\Gamma) \leqslant 0$.
If in 100 iterations the above stopping condition was not satisfied,it was then assumed that probably the global minimum had been found, and the algorithim was terminated. Obviously, this is only a necessary

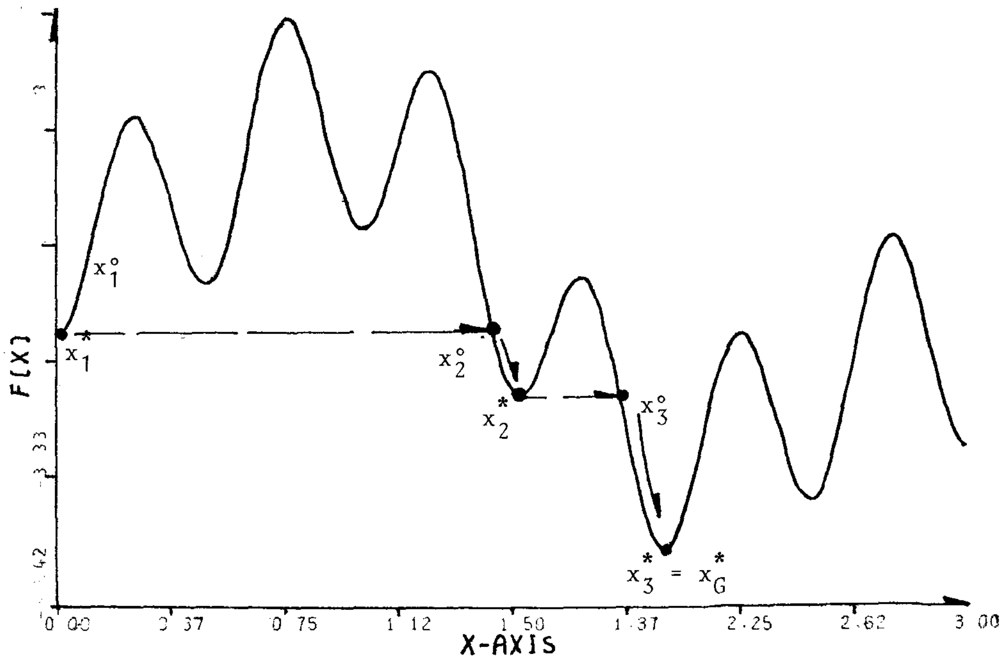

Fig.4a.Geometric interpretation of the Tunnelling Algorithm.The sequential use of Minimization phases and Tunnelling phases,makes the algorithm to ignore local minima whose function value is higher than the lowest f^* .

(a) $T(x,\Gamma) \leq 0$ at the level $f^* = 0$

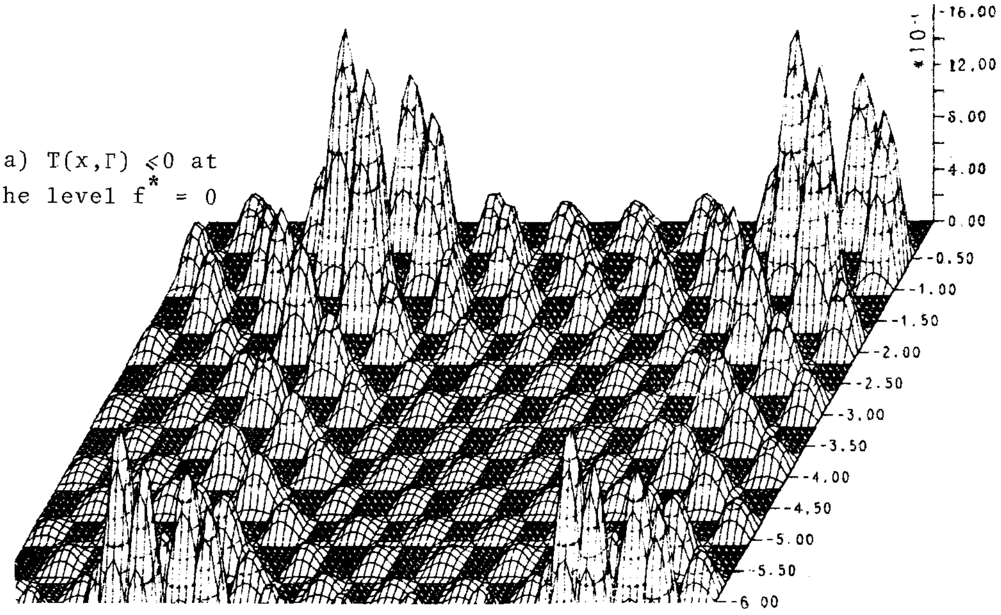

Fig.4b.Typical effect of storing f^* and shifting the original f(x) to obtain $f(x)-f^*$.Irrelevant local minima are ignored,regardless of their position.This figure is a partial illustration of Example 3, taking -f(x)/100 and f* =0

condition for global optimality.To obtain a necessary and sufficient condition for global optimality, instead of a 100 iterations,a very large computing time should be allowed to verify that $T(x,\Gamma) > 0$ for every x in the hypercube.

Each test problem was solved from N_R starting points, chosen at random. If each computer run took t_i CPU seconds to converge, an average computing time can be defined as

$$t_{av} = \frac{\sum_{i=1}^{N_R} t_i}{N_R} \qquad (9)$$

If N_S denotes the number of successful runs, the probability of success is given by

$$p = \frac{N_S}{N_R} \qquad (10)$$

Since the tunnelling phase starts from a random point near x^*, and finds one of the multiple points that can satisfy the stopping condition $T(x^\circ;\Gamma) \leqslant 0$,one can consider that the tunnelling phase behaves like a"random generator"of points x°.To see how well this"random generator" works, we shall generate the points $x^\circ$by other means,such as a random generator with a uniform probability distribution in the hypercube, and use the generated points as the nominals $x^\circ$of the next minimization phase.Two comparison algorithms are thus created,the Multiple Random Start and the Modified Multiple Random Start algorithms, as follows ;

MRS Algorithm : a)generate a random point x°. b)use this point as nominal for a minimization phase and find the corresponding x^*.c) Return to step a), unless the CPU computing time exceeds a given time limit, for example t_{av} of the tunnelling algorithm.

MMRS Algorithm : a)generate a random point x°.b) if $f(x^\circ)<f^*$ use x°as a nominal point for a minimization phase and find x^*, otherwise do not minimize and go back to step a). c)Stop when the CPU computing time exceeds a given time limit,for example the t_{av} of the tunnelling algorithm.

1.7 Conclusions

a). The tunnelling algorithm really "tunnels" below irrelevant local minima; from Fig. 5 we see that the number of local minima can go up by a factor of a million,from 10^3 to 10^8, while the computing effort goes up only by a factor of ten.

b). The computer runs show that the global minimum is approached in an

Ex. No.	No. Var.	No. Local Minima	TUNNELLING ALGORITHM				MRS		MMRS	
			N_f	N_g	t_{av}	p	t_{av}	p	t_{av}	p
1	1	3	758	129	0.55	1.0	0.09	1.0	0.60	0.5
2	1	19	1502	213	4.03	1.0	4.10	0.66	4.04	0.33
3	2	760	12160	1731	87.04	0.94	88.08	0.5	87.06	0.05
4	2	760	2912	390	8.47	1.0	5.19	- *	4.31	- *
5	2	760	2180	274	5.98	1.0	2.09	- *	6.01	0.0
6	2	6	1496	148	1.98	1.0	0.03	1.0	2.03	0.5
7	2	25	2443	416	3.28	1.0	64.0	- *	1.06	1.0
8	3	125	7325	1328	12.91	1.0	3.51	1.0	4.02	1.0
9	4	625	4881	1317	20.41	1.0	3.39	- *	4.33	1.0
10	5	10^5	7540	1122	11.88	1.0	8.10	- *	11.92	0.0
11	8	10^8	19366	2370	45.47	1.0	38.09	1.0	45.53	0.0
12	10	10^{10}	23982	3272	68.22	1.0	192.0	- *	68.26	0.0
13	2	900	2653	322	4.36	0.5	6.30	0.0	1.79	1.0
14	3	2700	6955	754	12.37	0.75	13.29	0.0	2.97	1.0
15	4	71000	3861	588	8.35	1.0	9.85	0.0	8.37	0.0
16	5	10^5	10715	1507	28.33	1.0	51.70	0.0	28.36	0.0
17	6	10^7	12786	1777	33.17	1.0	41.06	0.0	33.23	0.0
18	7	10^8	16063	2792	71.98	0.75	92.61	0.0	72.02	0.0

Fig. 5 Numerical Results. N_f= No. function eval. N_g= No. of gradient eval. t_{av}= average computing time (sec.). p= probability of success. * = Nonconvergence

orderly fashion, verifying in practice the generalized descent property of the tunnelling algorithm.

c). The tunnelling phase is a "very well educated random point generator"; for the same computing effort it provides points x_i^o that are better nominals for the next minimization phase, than those produced by the MRS and MMRS algorithms. This methods have a decreasing probability of success, $p \to 0$,as the density of local minima is increased.

d). For problems of small number of variables or with few local minima, a random nominal type algorithm requires less computing effort than the tunnelling algorithm; however, for problems with a large number of local minima, the tunneling algorithm is usually faster and has a higher probability of convergence.

e). Problems with several minima at the same function level are by far the most difficult to solve if all the "global minima" are desired. For instance , Example No. 3 has 18 "global minima" at the same level,(see

Figs. 2,4 and 5) and inspite of having only 760 local minima, the computing effort is similar to problems with 10^8 local minima.Examples No. 4 and 5 are obtained by from Ex. No.3, by removing 17 of the 18 "global minima",thus becoming much easier to solve.

1.8 Convergence Proof of a Modified Tunnelling Algorithm .

In Ref.5, a theoretical proof of the global convergence of a modified tunnellingalgorithm towards the global minimum of a one dimensional scalar function is given.
This version of the tunnelling algorithm, uses the same key ideas outlined in the previous sections, as well as the basic structure of sequential phases;a minimization phase and a tunnelling phase.The main modification is in the definition of the tunnelling function, it is defined as

$$T(x,\Gamma) = \frac{\{f(x)-f^*\}^2}{\{(x-x^*)^T(x-x^*)\}^\lambda} \tag{11}$$

The most important achievements of this modification are: a). It is now possible to establish the theoretical proof of the global convergence of the tunnelling algorithm to the global minimum of the function; thus it is guaranteed that, starting from any nominal point, the global minimum will be found in a finite number of steps. b). The proof of the convergence theorems are constructive proofs and therefore, a practical computer algorithm can be written that implements this global convergence properties. c). To experimentally validate the theoretical convergence theorems, this algorithm was written in FORTRAN IV in single precision and eight numerical examples of a single variable were solved in a B=6700 computer. The numerical results show that the global minimum was always found, regardless of the starting point, thus confirming the global convergence property of this modified tunnelling algorithm.

2. STABILIZED NEWTON'S METHOD. (Refs. 3,4)

2.1 Relationship to Global Unconstrained Minimization.

As described in section 1, during the tunnelling phase a zero of the tunnling function $T(x,\Gamma)$ must be found. The function $f(x)$ has usually many local minima. The introduction of the pole at x^* smooths out a few of them, but in general $T(x,\Gamma)$, will itself have many relative minima.
This means that the tunnelling function has many points x^s, where the gra-

dient $T_x(x,\Gamma)$ becomes

$$T_x(x,\Gamma) = 0$$

2.2 Statement of the Problem .

In this section we consider the problem of finding a zero of a system of non-linear equations $\Phi(x) = 0$, where Φ is a q-vector function with continuos first and second derivatives and x is an n-vector. We assume that there are many singular points x^s, where the Jacobian $\Phi_x(x^S)$ is not full rank and $\Phi(x) \neq 0$. We assume a solution exists.
In Ref.2 the particular case of q = 1 was studied with $A \leqslant x \leqslant B$, x an n-vector. In Ref.4 , the general case $q \leqslant n$ was presented; for brevity we shall only consider in this section the case q=n.
To clarify our notation, let the basic equation in Newton's method be written as

$$\Phi_x^T(x)\ \Delta x = -\Phi(x) \tag{12}$$

Solving this system of n equations gives the displacement vector Δx, and the next position vector is computed as

$$\tilde{x} = x + \beta\Delta x \qquad , \quad 0 < \beta \leqslant 1 \tag{13}$$

where the stepsize β is computed so as to enforce a descent property on the norm of the error, $P(x) = \Phi^T(x)\Phi(x)$, that is, starting from $\beta = 1$, β is bisected until the condition

$$P(\tilde{x}) < P(x) \tag{14}$$

is satisfied, accepting then $\tilde{x}$ as the new position vector to start the next solution of Eq. (12).

2.3 Description of the Stabilized Newton's Method .

It can be shown that, the damped Newton's Method is attracted to singular points, once it gets close to them, with the stepsize β tending to zero as the descent property on P(x) is enforced. Also as given in Refs. 3,4 it can be shown that, for the method to converge, it must be started close to the solution, having a small radius of convergence if singular points are present.
Design Goal. Since we want to solve systems of equations $\Phi(x) = 0$ that have many singular points, our goal is to design an algorithm that sta-

bilizes the Damped Newton's Method, increasing its convergence radius to infinity, making possible for the nominal point to be very far from the solution , even if many singular points are present.

Structure of the Stabilized Newton's Method . The design goal can be achieved as follows; a). detecting if the method is approaching a singular point and b). eliminating the singularity, so that the method is no longer attracted to it.

The stabilized algorithm has two possible phases; in one phase the zero of the original system $\Phi(x)$ is sought, however, when a singularity is detected the method will enter the other phase, were it will seek the zero of an equivalent system $S(x)=0$. For this structure to operate the system $S(x)$ must have the following properties;

$$\text{(a)}\quad \Phi(x^\circ)=0 \leftrightarrow S(x^\circ)=0 \qquad (15)$$
$$\text{(b)}\quad \Phi_x^{-1}(x^s) \text{ does not exist }, \quad S_x^{-1}(x^s) \text{ does exist}$$

and therefore even if $\Phi_x^{-1}(x^s)$ does not exist, the inverse $S_x^{-1}(x^s)$ does exist and we can still use a damped Newton's Method to find a zero of the equivalent system $S(x)=0$.

2.4 Derivation of the Equivalent System S(x).

Detection of the Singularity. If the damped Newton's Method applied to the original system $\Phi(x)$, generates points x_i , $i=1,2,...k$ which are being attracted to a singular point x^s, we know the stepsize β becomes very small; therefore if we detect that β becomes small in the bisection process, say smaller than $\beta<1/2^5$, we say that the present point x_k used in the algorithm is in the neighborhood of a singular point x^s.

Cancellation of the Singularity. Once we know the approximate position of the singular point, we define the equivalent system as follows

$$S(x) = \frac{\Phi(x)}{\{(x-x^m)^T(x-x^m)\}^\eta} \qquad (16)$$

where x^m denotes the position of a "movable pole" and η denotes its strength We define initially $x^m = x_k$ as the pole position, but later on it will be moved, hence its name. The rules to move it and recompute its pole strength are disscused in the following section.

2.5 Calculation of the Movable Pole Strength η .

This is easily done by the computer in the following automatic way;

Starting with $\eta = 0$ we compute the Jacobian

$$S_x(x) = \frac{1}{\{(x-x^m)^T(x-x^m)\}^\eta} \cdot \{\Phi_x(x) - 2\eta \frac{(x-x^m)\ \Phi^T(x)}{(x-x^m)^T(x-x^m)}\} \qquad (17)$$

at a point $x=x^m + \varepsilon$, where ε is a small random number, used just to avoid a division by zero in the computer. A tentative displacement Δx is computed from the system of equations

$$S_x^T(x)\ \Delta x = - S(x) \qquad (18)$$

and a new tentative position is obtained from $\tilde{x} = x + \beta\ \Delta x$.
If the Performance Index $Q(x) = S^T(x)S(x)$ does not satisfy the descent property $Q(\tilde{x}) < Q(x)$, we increase η to $\eta + \Delta\eta$ and repeat the above process. If the condition $Q(\tilde{x}) < Q(x)$ is satisfied we keep the present value of the pole strength η and $\tilde{x}$ as the new position vector for the next iteration on the equivalent system $S(x)$.

2.6 Calculation of the Movable Pole Position x^m.

The cancellation of the singularity is very effective with $x^m = x_k$ and η computed as in sec 2.5. A typical shape of $S(x)$ is shown in Fig. 6.a for $(x-x^m)^T(x-x^m) \leqslant 1$, and in Fig. 6.b for $(x-x^m)^T(x-x^m) > 1$. This last shape, pulse like and mostly flat, is produced by the scalar factor $1 / \{(x-x^m)^T(x-x^m)\}^\eta$ appearing both in the $S(x)$ and $S_x(x)$ equations, particularly if η is large, because the scalar factor makes $S(x) \to 0$ and $S_x(x) \to 0$, except in a small neighborhood of x^m. In general this shape causes Newton's Method to slow down, requiring a large number of iterations to find the zero of $S(x)$.
To avoid this unfavorable shape, we shall move the movable pole, whenever the condition $(x-x^m)^T(x-x^m) > R$ is satisfied, (say $R=1$). We will move the pole to the last acceptable position x_k, that is we set

$$x^m = x_k \quad , \eta = 0 \qquad (19)$$

and the pole strength η is recomputed for this new pole position as described in section 2.5 .

Summary of the Stabilized Newton's Method. A simplified flowchart of the method is given in Fig.7. We note a nice simple nested structure, unifying the structure of the Undamped, ($\eta=0$, $\beta=1$) , Damped, ($\eta=0$, $<\beta\leqslant 1$) , and the present Stabilized Newton's Methods, ($\eta \geqslant 0$, $<\beta\leqslant 1$).

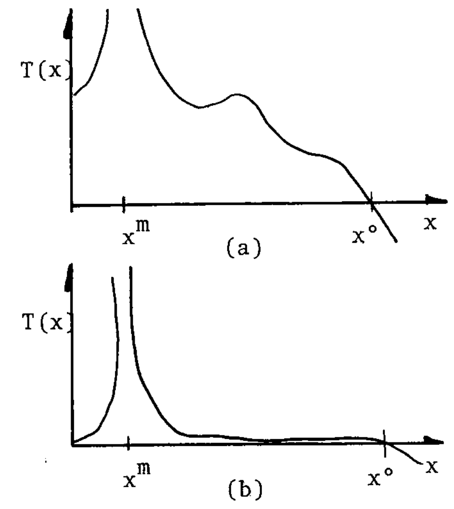

Fig. 6 Typical shapes of S(x).
(a).Favorable. $(x-x^m)^T(x-x^m)\leq 1$
(b). Unfavorable;$(x-x^m)^T(x-x^m)>1$

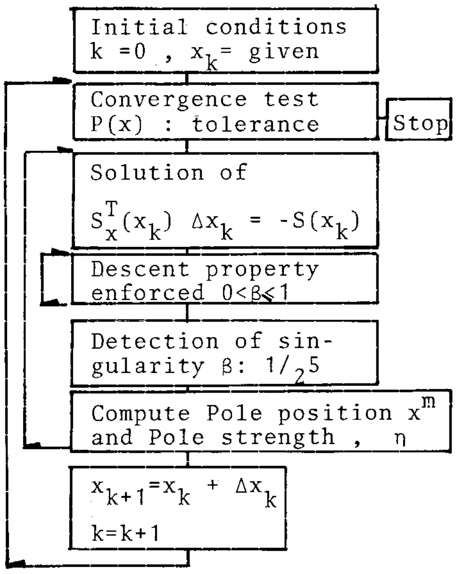

Fig. 7 Simplified Flowchart of the Stabilized Newton's Method, illustrating the nice,nested structure,unifying the Undamped,Damped and Stabilized Newton's Methods.

2.7 Numerical Examples.

A complete list of the examples can be found in Ref. 4. They vary from one equation to a system of five nonlinear equations.

Experimental conditions. The algorithm was programmed in FORTRAN IV in double precision, using a B-6800 computer.Three methods were compared Undamped,Damped and Stabilized Newton's Method.Each example was solved from 100 nominal points and the probability of succcess was computed as

$$p = \frac{N_S}{100} \qquad (20)$$

where N_S denotes the number of successful runs. If t_i denotes the CPU computing time required for each nominal to converge, then the average computing time is given by

$$t_{av} = \frac{\sum_{i=1}^{N_S} t_i}{N_S} \qquad (21)$$

Convergence was defined as $P(x) < 10^{-20}$ .Nonconvergence was assumed if the number of iterations exceded a given limit,for each nominal (say 500 iter.)

Ex. No.	UNDAMPED NEWTON		DAMPED NEWTON		STABILIZED NEWTON	
	p	t_{av}	p	t_{av}	p	t_{av}
1	0.78	1.29	0.72	0.18	1.00	0.48
2	0.67	0.37	0.98	0.10	1.00	0.22
3	0.81	0.47	1.00	0.08	1.00	0.16
4	0.56	0.41	0.53	0.69	0.81	6.60
5	0.42	1.46	0.79	0.27	1.00	0.26
6	0.67	0.41	0.94	0.28	1.00	0.56
7	0.98	0.92	0.84	0.23	1.00	1.08
8	0.44	0.07	0.78	0.33	0.99	0.84
9	0.49	0.24	0.50	0.15	0.96	1.92
10	0.41	3.97	0.24	0.92	0.63	3.80
11	0.76	10.20	0.21	0.96	0.97	7.47

Fig. 8 Numerical Results.
p = probability of success
t_{av} = Average computing time (sec.)

2.8 Conclusions .

The numerical results are shown in Fig. 8.This results indicate that ;
a). The Stabilized Newton's Method has a higher probability of converging to the solution,than the Undamped and Damped Newton's Method.If $\tilde{p}$ denotes the average over the 11 examples ,the Undamped has $\tilde{p}$ =0.64, the Damped has $\tilde{p}$ = 0.68 and the Stabilized Newton Method has $\tilde{p}$ =0.91 .We can say with confidence, that it is a Robust Method.
b). The convergence radius has been greatly increased; the nominals are very far away from the solution with many singular points.
c) The computing effort increases,due to the stabilization procedure in each iteration,however this increment is worthwhile and not exagerated, if $\tilde{t}_{av}$ denotes the average over the 11 examples, the Undamped has $\tilde{t}_{av}$= 1.80 sec. , the Damped has $\tilde{t}_{av}$ = 0.38 sec., and the Stabilized has $\tilde{t}_{av}$= 2.12 sec.

2.9 Use of the Stabilized Method in the Tunnelling Algorithm.

In order to find the zero of the Tunnelling Function, using the concept of the movable pole described in the Stabilized Newton's Method, the following expression is employed, instead of Eq.4

$$T(x,\Gamma) = \frac{f(x) \quad - \quad f(x^*)}{\{(x-x^m)^T(x-x^m)\}^\eta \; .\{(x-x^*)^T(x-x^*)\}^\lambda} \qquad (22)$$

where Γ denotes th set of parameters $\Gamma=(x^*,\lambda,x^m,\eta)$.$x^m$ and λ are updated at each iteration following the procedure for each parameter as indicated in the previous sections.

3. CONSTRAINED GLOBAL MINIMIZATION.

3.1 Statement of the Problem.

In this section the problem of finding the global minimum of f(x) subject to the constraints $g(x)\geqslant 0$ and $h(x)=0$ is considered,where f is a nonlinear scalar function, x is an n-vector, g and h are m and p non-linear functions defining the feasible set.Furthermore x is limited to the hypercube $A\leqslant x\leqslant B$ where A and B are prescribed n-vectors.
The feasible set can be a convex set or the constraints may defined several non-convex,non-conected feasible sets.Due to the importance of this topic, it is presented in a companion paper "Global Optimization of Functions Subject to Non-convex,Non-connected Regions." by Levy and Gomez.

REFERENCES

1.A. V. Levy and A. Montalvo, " TheTunnelling Algorithm for the Global Minimization of Functions ", Dundee Biennal Conference on Numerical Analysis,Univ. of Dundee,Scotland,1977.
2.A.V. Levy and A. Montalvo,"Algoritmo de Tunelizacion para la Optimacion Global de Funciones", Comunicaciones Tecnicas, Serie Naranja,No. 204, IIMAS-UNAM,Mexico D.F.,1979.
3. A.V. Levy and A. Calderon, "A Robust Algorithm for Solving Systems of Non-linear Equations", Dundee Biennal Conference on Numerical Analysis, Univ. of Dundee,Scotland,1979.
4. A. Calderon,"Estabilizacion del Metodo de Newton para la Solucion de Sistemas de Ecuaciones No-Lineales" ,B. Sc. Thesis, Mathematics Dept. UNAM, Mexico D.F. ,1978.
5. A.V. Levy and A. Montalvo," A Modification to the Tunnelling Algorithm for Finding the Global Minima of an Arbitrary One Dimensional Function ", Comunicaciones Tecnicas,Serie Naranja,No. 240,IIMAS-UNAM,1980.
6. D. M. Himmelblau, "Applied Nonlinear Programing",Mc Graw-Hill Book Co. N.Y. ,1972.

APENDIX : TEST PROBLEMS

Example 1.

$$f(x) = x^6 - 15x^4 + 27x^2 + 250$$

$$-4 \leqslant x \leqslant 4$$

Example 2.

$$f(x) = -\sum_{i=1}^{5} i\cos\{(i+1)x + i\}$$

$$-10 \leqslant x \leqslant 10$$

Example 3.

$$f(x,y) = \left(\sum_{i=1}^{5} i\cos\{(i+1)x+i\}\right) \left(\sum_{i=1}^{5} i\cos\{(i+1)y+i\}\right) + \beta\{(x + 1.42513)^2 + (y + 0.80032)^2\}$$

$$-10 \leqslant x,y \leqslant 10 , \quad \beta = 0.$$

Example 4.

See Example 3., with $\beta = 0.5$

Example 5.

See Example 3. , with $\beta = 1.0$

Example 6.

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

$$-3 \leqslant x \leqslant 3 , \qquad -2 \leqslant y \leqslant 2$$

Example 7

$$f(x) = \pi\{k\sin^2\pi y_1 + \sum_{i=1}^{n-1}\{(y_i - A)^2(1 + k\sin^2\pi y_{i+1})\} + (y_n - A)^2\}/n$$

with A=1 ,k=10 , n=2 and $y_i = 1 + (x_i - 1)/4$

$$-10 \leqslant x_i \leqslant 10, \; i = 1,2,\dots,n$$

Examples 8,9,10,11,12, See Ex. 7 with n=3,4,5,8,10 ,respectively.

Example 13.

$$f(x) = k_1\sin^2\pi l_0 x_1 + k_1\sum_{i=1}^{n-1}\{(x_i - A)^2(1 + k_0\sin^2\pi l_0 x_{i+1})\} + k_1(x_n - A)^2(1 + k_0\sin^2\pi l_1 x_n)$$

$A = 1,\ l_0 = 3$ y $l_1 = 2.\ k_0 = 1,\ k_1 = 0.1\ n = 2,\ -10 \leqslant x_i \leqslant 10$

Examples 14,15, See Ex. 13 with $-10 \leqslant x_i \leqslant 10$ and n=3,4 respectively
Examples 16,17,18, See Ex. 13 with $-5 \leqslant x_i \leqslant 5$ and n=5,6,7 resp.