TRUST: A New Global Optimization Methodology, Application to Artificial Neural Networks, and Analog VLSI Implementation

Thesis by Bedri Çağ Çetin

In Partial Fulfillment of the Requirements
for the Degree of
Doctor of Philosophy

California Institute of Technology
Pasadena, California
1994

(defended on July 20, 1993)

Acknowledgements

This thesis would never have seen the light without the collaboration and help of my advisor, Joel Burdick. Thanks to Joel for his invaluable support, advice and ideas, which inspired this work. Special thanks to Jacob Barhen, who has been a second advisor as well as a frequent and important collaborator at Jet Propulsion Laboratory.

Many thanks to Doug Kerns for help designing and testing the chips. I am also indebted to Rahul Sarpeshkar, who gave significant advice on designing the control logic circuit.

I am deeply indebted to my parents, Basri and İsmet Çetin for their extraordinary support, encouragement and concern. I am very grateful to my wife Renée for her moral support, love and patience.

This work was supported in part by the Department of Energy, Office of Basic Energy Sciences (Grant # DE-AI05-89 ER14086). Additional support was provided by JPL Director's Discretionary Funds and National Science Foundation (Grant # MSS-9157843). All opinions, findings and conclusions, or recommendations expressed in this document are those of the author and do not necessarily reflect the views of the sponsoring agencies.

Abstract

A new method for unconstrained global function optimization, acronymed TRUST, is introduced. This method formulates optimization as the solution of a deterministic dynamical system incorporating terminal repellers and a novel subenergy tunneling function. Benchmark tests comparing this method to other global optimization procedures are presented, and the TRUST algorithm is shown to be substantially faster.

This algorithm is provably convergent to the global minimum for objective functions of one variable. Theoretically, convergence to a global solution is not guaranteed in the multi-dimensional case. However, in practical applications, TRUST has found the global minimum in all multi-dimensional benchmark functions as a result of its global descent property. The TRUST formulation leads to a simple stopping criterion.

The algorithm is also applied to Backpropagation learning in artificial neural networks in order to overcome the susceptibility to local minima during training, which is associated with gradient descent. TRUST (or Global Descent in this context) was proposed as a candidate for replacing gradient descent in order to eliminate the local minima problem. We test the ability of the new dynamical system to overcome local minima with common benchmark examples and a pattern recognition example. The results demonstrate that the new method does indeed escape encountered local minima, and in most cases converges to the globally optimal solution of a specific problem.

The structure of the TRUST's equations enables an implementation of the algorithm in analog VLSI hardware for further substantial speed enhancement. We have designed, fabricated and tested a terminal repeller circuit and a gradient descent circuit, which constitute the main components of the TRUST's dynamics. Measured chip data, which confirmed the efficient performance of these circuits, are presented. We have also designed a novel global optimization circuit which incorporates the above circuits with additional control logic. This circuit implements the TRUST algorithm, and thus locates the global minimum of arbitrary one-dimensional objective functions. Simulated experiments of this circuit are thoroughly discussed. The convergence time required for the circuit to converge to the global minimum is remarkably at the order of micro-seconds.

Contents

Chapter 1

Introduction

1.1 TRUST Algorithm

Many engineering applications can be formulated as a nonlinear function optimization problem in which the function to be optimized (i.e., objective function) possesses many local minima in the parameter region of interest. In most cases, it is desired to find the local minimum at which the function takes its lowest value, i.e., the global minimum. The problem of designing algorithms that can distinguish between the global minimum and the numerous local minima is known as the *global optimization problem.*

In the past few years, several methodologies related to global optimization have been developed and investigated. These algorithms can be divided into two primary classes: probabilistic and deterministic.

Most probablistic methods involve the evaluation of the objective function at a random sample of points which are drawn from a uniform distribution over the domain of interest. Following this, some probabilistic measures are applied to the samples to determine the smaller region where there exist a higher probability for the global minimum. The simplest probabilistic algorithm is the Pure Random Search Algorithm by Brooks [Bro58] and Anderssen [And72], where a single local search is performed starting from the best point in the sample set. Here, the best point corresponds to the sample that has the highest probability

to be in the region of the global minimum. An extension of this algorithm is the Multiple Random Search Method by Bremermann [Bre70]. In this method each random point is taken as a starting point for a local optimization and the global minimum is identified by comparing the corresponding results.

Another widely used probabilistic method is a Monte Carlo technique called Simulated Annealing (SA) [KGV83] which is analogous to the annealing process of metals. Simulated annealing emulates the process by which a molten metal is slowly cooled to reach its crystalline form, which is the lowest ground state, or global minimum, of the material's energy. The cooling schedule employs Gaussian random processing which is inversely proportional to the logarithm of time. The evolution of the process in time is performed according to a probabilistic measure based on Boltzman distribution. Theoretically, it has been proven that the global minimum of a cost function can be reached as long as the cooling schedule is sufficiently slow [MRTT53]. Therefore the choice of a cooling schedule is critical to the performance of this technique. Szu and Hartley [SH87] introduced a related algorithm termed Fast Simulated Annealing (FSA), which is a modification of SA. This algorithm employs a faster cooling schedule, which is based on Cauchy random processing, that is inversely proportional of time.

While SA and FSA have received a wide following, other stochastic global optimization methods exist. An extensive review of such computational schemes can be found in [KT85]. The algorithms mentioned above and all other probabilistic methods developed so far have their own advantages and disadvantages, but since they rely on stochastic processes, they can require formidable computational effort to reach a global minimum.

A number of deterministic methods exist in the literature. Most of these techniques are tunneling based approaches. Generally tunneling methods, as their name implies, rely upon the idea of finding a new optimization initial condition in another "lower valley" in order to escape from the current local minimum and to continue the optimization procedure which will lead to the global minimum. The Tunneling Algorithm introduced by Levy and Montalvo [LM85], and the Dynamic Tunneling Algorithm by Yao [Yao89] are thoroughly

reviewed in Section 2.2.

This thesis presents a new global optimization scheme whose acronym is **TRUST** (*Terminal Repeller Unconstrained Subenergy Tunneling*). This method incorporates a tunneling feature, which is substantially different than other tunneling based techniques. In this approach, we formulate optimization as the solution to a deterministic dynamical system which incorporates a novel subenergy tunneling functional and terminal repellers. In addition, the TRUST formulation leads to a well-defined stopping criterion. Chapter 2 introduces the TRUST algorithm and compares TRUST performance to other global optimization methods. Several standard one- and multi-dimensional objective functions taken from the literature are used as benchmark functions. The standard benchmark tests demonstrate that TRUST is significantly faster than previously published techniques. The simulation parameters are listed in Appendix A.

The TRUST computational scheme can be guaranteed to find the global minimum for functions of one variable. The method is currently not guaranteed to find the global minima in multiple dimensions. However, in practice, as a result of its global descent property, the global minimum was found in all benchmark simulations, including 10-dimensional test functions. Furthermore, the structure of the optimizing dynamical system is highly parallel, allowing implementation in a form whose computational complexity is only weakly dependent on problem dimensionality.

1.2 Local Minima Problem in Artificial Neural Networks

Neural networks with massively parallel processing units provide an effective approach for a broad spectrum of applications – pattern mapping, pattern completion and pattern classification. In this thesis, emphasis is given on those tasks of the artificial neural networks where the network maps a set of input patterns to a set of output patterns (i.e., function learning tasks). One of the most influential developments in this area was the invention of the Backpropagation algorithm [RHW86], which is a systematic method for training multilayer artificial neural networks.

Backpropagation is a learning algorithm that gives a prescription for changing the connection weights of a feed-forward network to learn a training set of input-output pairs without any prior knowledge of the mathematical function that maps them. Given an error energy measure [E] (e.g., summation of the squared differences between the actual and target output values), backpropagation uses gradient descent to adjust the weights following the local slope of the error surface towards a minimum. Thus, it is hoped to find a global minimum to the error energy surface, which will correspond to the optimal weights that solve the specific mapping problem. However, gradient descent is a dynamical system which only locally minimizes the error energy function.

Despite its popularity, Backpropagation learning by gradient descent has two major drawbacks — the slow convergence time and the presence of local minima. First, there is no guarantee that the network can be trained in a reasonable amount of time because the convergence process may be exceedingly long. Second, there is no assurance that the network will train to the best configuration possible, since a local minimum found by gradient descent can trap the training algorithm in an inferior solution, whereas ideally a global minimum is desired to solve the problem.

Statistical training methods have previously been proposed to alleviate the local minima problem. Hinton and Sejnowski [HS86] introduced the Boltzmann Machine Learning Algorithm, which employs a simulated annealing type of approach in order to control the connection weights of the network. Here, upward steps in E are allowed occasionally according to the Boltzman probability distribution, with a temperature T controlling the probabilities. Annealing—a gradual lowering of T—is then performed. Such upward motion in E is expected to bring the states of the network out of the local minima. This technique has been extended by Wasserman [Was89], who suggested the employment of simulated annealing in Backpropagation learning. Thus, backpropagation is modified to have two components: a deterministic component due to gradient descent; and a random component determined by simulated annealing. Alternatively, von Lehman et al. [vLPL+88] proposed to add noise explicitly by randomly changing the connection weights of the network slightly. Sietsma and

Dow [SD88] offered to add the noise to the training set inputs instead. All these methods aim to alleviate the local minima problem in the context of learning with neural networks, but suffer from extreme slow convergence due to their probabilistic nature.

Chapter 3 discusses primarily the local minima problem associated with gradient descent and proposes the use of the TRUST algorithm to overcome it. In the context of artificial neural networks, TRUST (hereafter named Global Descent) provides a simple extension to the Backpropagation algorithm by replacing the gradient descent method during training. Global Descent is a deterministic method and therefore is not limited by the drawbacks (e.g., the slow convergence time) of the probabilistic techniques mentioned above.

The new formalism has been tested for common benchmarks, like the XOR and parity functions, and also for a pattern recognition example using 60 patterns. The results demonstrate that Backpropagation associated with Global Descent escapes encountered local minima and in most cases converges to the globally minimal solution. Chapter 3 also presents these simulation results.

1.3 Implementing TRUST

The deterministic dynamics of TRUST enables an implementation of the algorithm in analog VLSI circuitry for further speed enhancements. In order to implement the TRUST formalism, we first implemented its main components. A terminal repeller circuit and a gradient descent circuit have been designed and fabricated for this purpose (Chapter 4).

Zak [Zak89] has introduced terminal dynamical systems, which are based on the non-Lipschitzian dynamics of odd power-law transfer characteristics. As a result of this property, terminal repellers form systems whose dynamics will be repelled from the unstable point (i.e., the repeller) in a finite amount of time. Similarly, terminal attractors constitute systems whose dynamics will be attracted by the stable point in finite time. Hence, terminal systems are important tools of nonlinear dynamical systems. Zak, Barhen, and Toomarian [Zak89, ZB90, BTG90, BZT90b, BZT90a] have used the concept of terminal attractors and repellers in the context of neural network dynamics to obviate the infinite-time solution

limitations of regular attractors and repellers. In the context of global optimization, terminal repellers allow the TRUST dynamics to be repelled from the local minimum at high speeds. Terminal systems are discussed in Subsections 2.3.2 and 2.3.3 in further detail.

To the best of out knowledge, there exist no implementation of terminal systems in the literature. We have designed analog VLSI circuits that implement terminal attractors and terminal repellers. The terminal repeller circuit required for the TRUST algorithm has been fabricated and tested. Section 4.1 describes the research and experiments we have done, including our measured chip results. The results demonstrate that almost ideal terminal system performance has been achieved.

The second main part of the TRUST formulation is gradient descent. Gradient descent is a dynamical system, whose stable equilibrium point locally minimizes a given objective function. Thus, it is useful for a wide range of optimization applications and its role in learning with artificial neural networks is discussed in Section 3.1. Gradient descent enables TRUST dynamics to locally minimize the objective function as soon as a functionally lower valley is reached by the effect of the terminal repeller mentioned above. A wider description of the role of gradient descent in TRUST formalism is given in Subsection 2.3.3.

Section 4.2 presents the gradient descent circuit which we have developed in analog VLSI hardware. This circuit approximates the gradient descent dynamics by measuring the gradient based on discrete samples of the objective function. It is shown the approximation is valid as long as the sampling interval is sufficiently small (see Section 4.2 for corresponding discussion). The measured chip results corresponding to different experiments testing this circuit are also given. Our findings show that the gradient circuit performs stably and relatively accurately (except for very steep functions) for locating the critical points of arbitrary one-dimensional objective functions.

Similar work independently has been done by Umminger and DeWeerth [UD89], who uses the same conceptual approach as ours with a different circuit design. In order to enhance the accuracy of the gradient operation, an alternative method has been proposed by Anderson et al. [AK]. Their model estimates the gradient of the function by using a

noise-function correlation approach. Kirk [Kir93] has further extended this approach to handle multiple dimensions. A detailed discussion of this technique can be found in [Kir93].

By employing these circuits—terminal repeller and gradient descent, and additional control logic circuit, we have designed a global optimization circuit, which is the subject of Section 4.3. This circuit implements one-dimensional TRUST dynamics. A thorough analysis and discussion of this circuit together with its simulated experiments are also described in Section 4.3. It is shown that the circuit performs very efficient global optimization of arbitrary one-dimensional functions. The convergence time required for the circuit to converge to the global minimum is remarkably at the order of micro-seconds.

Chapter 2

TRUST for Fast Global Optimization

2.1 Problem Formulation

The global optimization problem to be considered in this thesis can be stated as follows. Let $f(\vec{x}) : \mathcal{R}^n \rightarrow \mathcal{R}$ be a twice continuously differentiable function, where $\vec{x}$ is a vector of n state variables or parameters. Hereafter, $f(\vec{x})$ will be referred to as the *objective* function. The goal is to find the value, $\vec{x}_{GM}$, of the state variables which minimizes $f(\vec{x})$,

$$f^* = f(\vec{x}_{GM}) = \min\{f(\vec{x}) |\ \vec{x} \in \mathcal{D}\}, \tag{2.1}$$

where $\mathcal{D}$ is the domain of interest over which one seeks the global minimum. $\mathcal{D}$ is assumed to be compact and connected. In the sequel, and without loss of generality, we assume $\mathcal{D}$ to be the hyper-parallelpiped

$$\mathcal{D} = \{x_j |\ x_{jL} \leq x_j \leq x_{jU}; \quad j = 1, 2, \cdots, n\}, \tag{2.2}$$

where x_{jL} and x_{jU} are respectively the lower and upper bounds on the jth state variable. The compactness of $\mathcal{D}$ and continuity of $f(\vec{x})$ ensure that $f(\vec{x})$ is bounded away from infinite

magnitude in the domain of interest. Further, we assume that every local minimum $\vec{x}_{LM}$ of $f(\vec{x})$ in $\mathcal{D}$ satisfies the conditions

$$\frac{\partial f(\vec{x}_{LM})}{\partial \vec{x}} = 0, \tag{2.3}$$

$$\vec{y}^T \frac{\partial^2 f(\vec{x}_{LM})}{\partial \vec{x}^2} \vec{y} \geq 0, \qquad \forall \vec{y} \in \mathcal{R}^n. \tag{2.4}$$

We further assume that the global minimum satisfies these local minimum criteria and that the global minimum does NOT occur on the boundary of $\mathcal{D}$.

Section 2.2 reviews previous global optimization approaches which are relevant to this work. This review focuses on tunneling methods, since the TRUST algorithm introduces a novel approach to tunneling. Section 2.3 presents the one-dimensional TRUST optimization algorithm. Section 2.4 discusses the convergence properties of the one-dimensional algorithm, while Section 2.5 considers the multi-dimensional TRUST scheme. Section 2.6 presents the results of benchmark simulations and compares the TRUST performance to other global optimization methods. Section 2.7 summarizes Chapter 2.

2.2 Methodologies for Global Optimization: Background

Previously developed global optimization algorithms can be roughly categorized into two classes: probabilistic and deterministic. An extensive review of probabilistic computational schemes can be found in [KT85]. Here we focus on deterministic tunneling methods, as these are most closely related to the concept presented in this thesis.

Tunneling for global optimization was introduced by Levy and Montalvo [LM85]. Their tunneling method is composed of a sequence of cycles, where each cycle has two phases: a local minimization phase and a tunneling phase. In the first phase, minimization algorithms such as gradient descent or Newton's method are employed to minimize $f(\vec{x})$. We assume that starting from an initial point $\vec{x}^{0(0)}$, the minimization converges to the first local minimum $\vec{x}^{1(*)}$, which satisfies conditions (2.3) and (2.4).

In the second phase, a tunneling function is defined,

$$T(\vec{x}, \vec{x}^{1(*)}) = \frac{\hat{f}(\vec{x})}{[(\vec{x} - \vec{x}^{1(*)})^T (\vec{x} - \vec{x}^{1(*)})]^{\alpha}} , \tag{2.5}$$

where

$$\hat{f}(\vec{x}) = f(\vec{x}) - f(\vec{x}^{1(*)}). \tag{2.6}$$

The tunneling phase searches for the zeros of $T(\vec{x}, \vec{x}^{1(*)})$; that is, $T(\vec{x}, \vec{x}^{1(*)}) = 0$ is solved for any $\vec{x}^{1(0)}$ such that $\vec{x}^{1(0)} \neq \vec{x}^{1(*)}$, but $f(\vec{x}^{1(0)}) = f(\vec{x}^{1(*)})$. The denominator of (2.5) is a pole of strength α located at the previously determined local minimum $\vec{x}^{1(*)}$, thus preventing the zero-finding algorithm from rediscovering $\vec{x}^{1(*)}$ as a zero of the tunneling function. The zero $\vec{x}^{1(0)}$ of (2.5) is used as the starting point of the next cycle, and the process is repeated sequentially, as shown in Figure 2.1, until a stopping criterion, such as the failure to find a zero within a prescribed CPU time, is met. The last local minimum to be found is assumed to be the global minimum.

If we denote $\vec{x}^{i(*)}$ as the minimum reached during the ith minimization phase, the tunneling algorithm implements a global descent property,

$$f(\vec{x}^{i+1(*)}) \leq f(\vec{x}^{i(*)}).$$

However, this method has a number of disadvantages:

(i) The pole strength α is problem dependent. While searching for a zero, α should be incrementally increased until the pole in the denominator of (2.5) becomes strong enough to eliminate the last local minimum of higher order. Every increase in α requires the algorithm to be restarted, leading to increased computational effort.

(ii) The tunneling algorithm may find another local minimum $\vec{x}^{2(*)}$, such that $f(\vec{x}^{1(*)}) = f(\vec{x}^{2(*)})$. In this case, an additional pole must be placed at the second local minimum, and the tunneling process must be restarted.

(iii) Division by a pole causes smoothing of $f(\vec{x})$ as $\vec{x} \to \infty$; that is, $f(\vec{x}) \to 0$ as $\vec{x} \to \infty$.

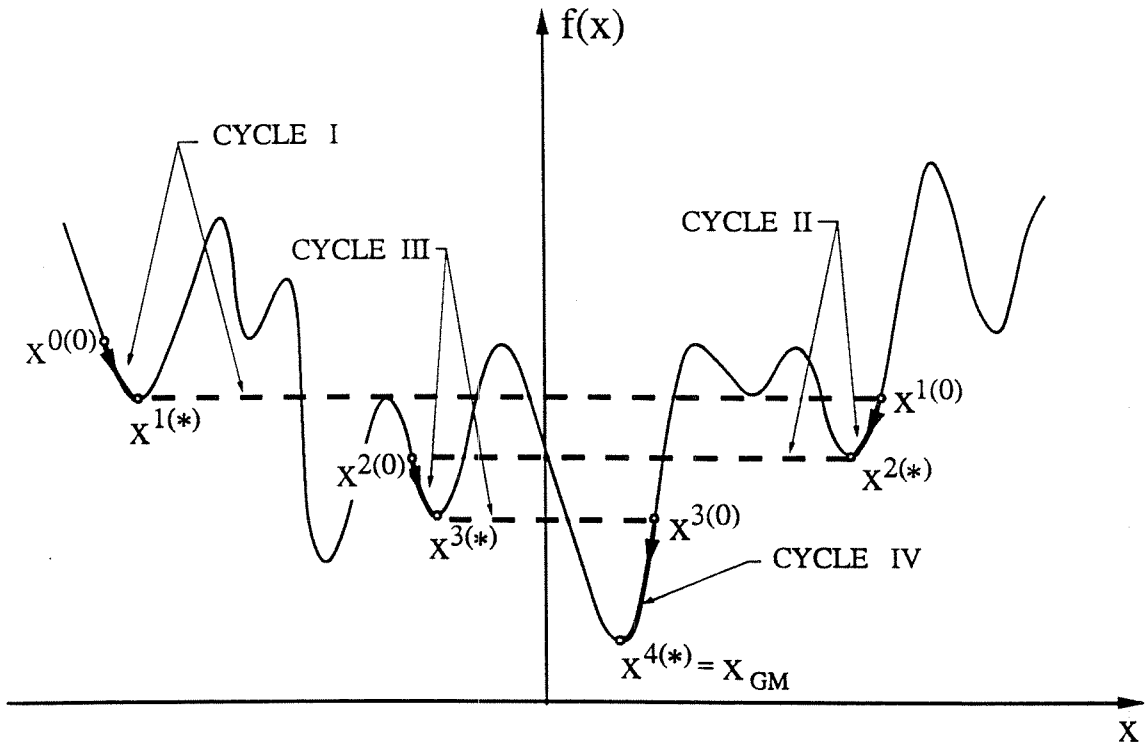

Figure 2.1: Schematic diagram of tunneling operation.

This smoothing increases with α, yielding a tunneling function that becomes very flat. In this case, zeros can be difficult to detect correctly.

(iv) The zero-finding algorithm in [LM85] is based on a modified Newton iteration which requires finding the roots of a scalar function with multiple variables. This can be a computationally expensive procedure, and as yet there are no globally convergent zero finding algorithms. Thus, stopping criteria cannot easily be defined.

The difficulties associated with finding the zeros of (2.5) have been partly overcome by the dynamical tunneling algorithm of Yao [Yao89]. His dynamical tunneling procedure has two phases: dynamic optimization and dynamic tunneling. The dynamic optimization

phase implements minimization via gradient descent,

$$\dot{\vec{x}} = -\frac{\partial f(\vec{x})}{\partial \vec{x}}. \tag{2.7}$$

Starting from an initial point $\vec{x}^{0(0)}$, the system (2.7) reaches its first equilibrium at a local minimum $\vec{x}^{1(*)}$. However, in the second phase, instead of finding the zeros of the tunneling function (2.5), Yao defines an energy function

$$E(\vec{x}, \vec{x}^{1(*)}) = T(\vec{x}, \vec{x}^{1(*)}) + k \int_0^{\hat{f}(\vec{x})} zu(z)dz, \tag{2.8}$$

where $u(z)$ is the Heaviside step function,

$$u(z) = \begin{cases} 1, & z \geq 0, \\ 0, & z < 0. \end{cases} \tag{2.9}$$

The energy function in (2.8) is minimized in Yao's tunneling phase, instead of finding the zeros of $T(\vec{x}, \vec{x}^{1(*)})$. The derivative of (2.8) with respect to its state vector $\vec{x}$ is:

$$\begin{aligned} \frac{\partial E(\vec{x}, \vec{x}^{1(*)})}{\partial \vec{x}} &= \frac{(\partial f/\partial \vec{x})\,||\vec{x} - \vec{x}^{1(*)}||^{2\alpha} - 2\alpha(\vec{x} - \vec{x}^{1(*)})\,||\vec{x} - \vec{x}^{1(*)}||^{2(\alpha-1)}\hat{f}(\vec{x})}{||\vec{x} - \vec{x}^{1(*)}||^{4\alpha}} \\ &+ k\frac{\partial f(\vec{x})}{\partial \vec{x}}\hat{f}(\vec{x})u(\hat{f}(\vec{x})). \end{aligned} \tag{2.10}$$

From (2.10), it is clear that the second term in (2.8) enforces the constraint $\hat{f}(\vec{x}) \leq 0$ [i.e., $f(\vec{x}) \leq f(\vec{x}^{1(*)})$] if the magnitude of k is chosen large enough. When gradient descent is applied to $E(\vec{x}, \vec{x}^{1(*)})$ in (2.8), we obtain the dynamical system

$$\dot{\vec{x}} = -\frac{\partial E(\vec{x}, \vec{x}^{1(*)})}{\partial \vec{x}}. \tag{2.11}$$

The initial conditions for this system are $\vec{x}^{1(*)} + \vec{\epsilon}$, where $\vec{\epsilon}$ is a small perturbation which displaces the system from the tunneling function pole located at $\vec{x}^{1(*)}$. When (2.11) converges to its final equilibrium state, it minimizes the tunneling function with respect to the

constraint $\hat{f}(\vec{x}) \leq 0$. Thus, the system in (2.11) will reach an equilibrium point $\vec{x}^{1(0)}$ that lies in another basin of attraction, with functional values lower than $f(\vec{x}^{1(*)})$, if one exists. This new equilibrium point will be the starting point for the dynamic optimization phase of the next cycle. The procedure is repeated until a new equilibrium in a lower valley can not be found in a prescribed amount of time. It is then assumed that the last minimum is the global minimum.

This approach also has a number of deficiencies:

(i) The pole-strength α must be chosen sufficiently high to enable the pole in the denominator of (2.5) to cancel the last local minimum of higher order, and thereby prevent restarting of the tunneling phase, as this necessitates back tracking of (2.11).

(ii) The penalty constant k is problem dependent, and a global minimum cannot be guaranteed for a prescribed k.

(iii) An implementation of global optimization in terms of the solution of two different dynamical systems in two different phases makes the algorithm impractical for implementation in analog VLSI hardware. A method based on a single differential equation would be preferable.

In this thesis, we introduce a deterministic global optimization methodology which is also based upon the concept of dynamic tunneling. However, in contrast to these previous approaches, tunneling is implemented here in a substantially different manner, by employing so-called terminal repellers and a novel subenergy tunneling function. The next section introduces these concepts and assembles them into an optimization algorithm which is the solution of a single vector differential equation. This characteristic simplifies the hardware implementation of our algorithm.

2.3 Terminal Repeller Unconstrained Subenergy Tunneling Algorithm

2.3.1 Subenergy Tunneling Function

We define a *subenergy tunneling function*, or *subenergy function* for short, as follows:

$$E_{sub}(\vec{x}, \vec{x}^*) = \log\left(\frac{1}{1 + exp(-(\hat{f}(\vec{x}) + a))}\right), \tag{2.12}$$

where

$$\hat{f}(\vec{x}) = f(\vec{x}) - f(\vec{x}^*) \tag{2.13}$$

and a is a constant whose value will be considered below. In the above expression $\vec{x}^*$ is a fixed value of $\vec{x}$, whose selection will be discussed in Subsection 2.3.4.

Equation (2.12) is a nonlinear but monotonic transformation of $f(\vec{x})$ which has several useful properties. First, the derivative of $E_{sub}(\vec{x}, \vec{x}^*)$ with respect to $\vec{x}$ is:

$$\frac{\partial E_{sub}(\vec{x}, \vec{x}^*)}{\partial \vec{x}} = \frac{\partial f(\vec{x})}{\partial \vec{x}} \frac{1}{1 + exp(\hat{f}(\vec{x}) + a)}. \tag{2.14}$$

Since

$$\frac{1}{1 + exp(\hat{f}(\vec{x}) + a)} > 0, \qquad \forall\ \vec{x} \in \mathcal{D},$$

we conclude that

$$\frac{\partial E_{sub}(\vec{x}, \vec{x}^*)}{\partial \vec{x}} = 0 \quad \Leftrightarrow \quad \frac{\partial f(\vec{x})}{\partial \vec{x}} = 0. \tag{2.15}$$

From (2.15), it is clear that $E_{sub}(\vec{x}, \vec{x}^*)$ has the same critical points as $f(\vec{x})$ and the same relative ordering of the local and global minima. In other words, $E_{sub}(\vec{x}, \vec{x}^*)$ is a transformation of $f(\vec{x})$ which preserves all properties relevant for optimization. In addition, this transformation is intended to have the following effect. We wish $E_{sub}(\vec{x}, \vec{x}^*)$ to asymptotically but quickly approach zero for $\hat{f}(\vec{x}) \geq 0$. Second, we would like to leave $\hat{f}(\vec{x})$ nearly unmodified for $\hat{f}(\vec{x}) < 0$. Hereafter, $f(\vec{x}^*)$ will be referred to as the *zero subenergy limit*,

since

$$E_{sub}(\vec{x}, \vec{x}^*) \simeq 0, \qquad \text{for } f(\vec{x}) \geq f(\vec{x}^*).$$

The monotonicity of the transformation is *not* affected by the particular value of the constant a, though the asymptotic properties *are* affected by its value. Figure 2.2 plots $E_{sub}(\vec{x}, \vec{x}^*)$ vs $\hat{f}(\vec{x})$ for various values of a. The algorithm can be formulated to work for nearly any reasonable value of this parameter. In subsequent analyses, the necessary and sufficient values of other TRUST algorithm parameters are derived in terms of a. However, for practical applications, a value $a = 2$ is chosen, as it leads to the most desirable asymptotic behavior of the subenergy tunneling transformation.

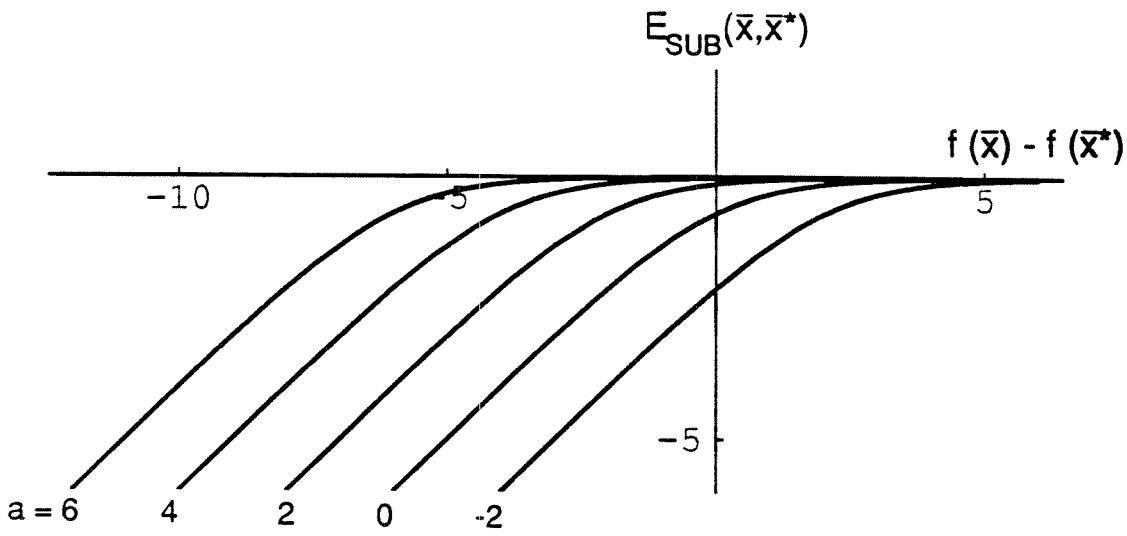

Figure 2.2: Behavior of $E_{sub}(\vec{x}, \vec{x}^*)$ vs $\hat{f}(\vec{x})$ for various values of a.

Figure 2.3 shows an example of a one-dimensional function,

$$f(x) = (\sin(2x) - x - 1)^2,$$

to which the transformation in (2.12) has been applied for the case

$$x^* = -6.80678, \quad a = 2.$$

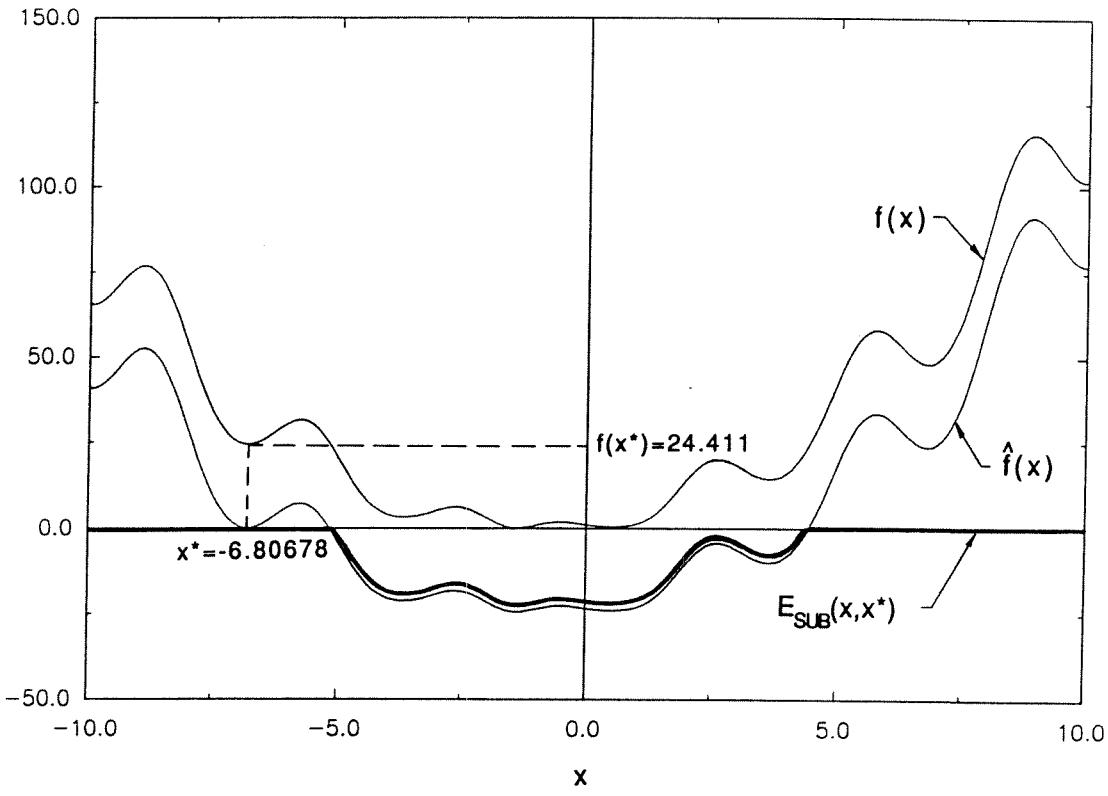

Figure 2.3: Example of one-dimensional subenergy tunneling transformation.

Figure 2.4B shows an example of the transformation $E_{sub}(\vec{x}, \vec{x}^*)$ applied to the two-dimensional function (Figure 2.4A):

$$f(x, y) = (x - 0.1)^2(y - 0.2)^2 + 3\sin(0.2 + 1.5\pi x^2)\sin(0.3 + \pi y)$$

for the case

$$(x^*, y^*) = (0, -3/2), \qquad a = 2.$$

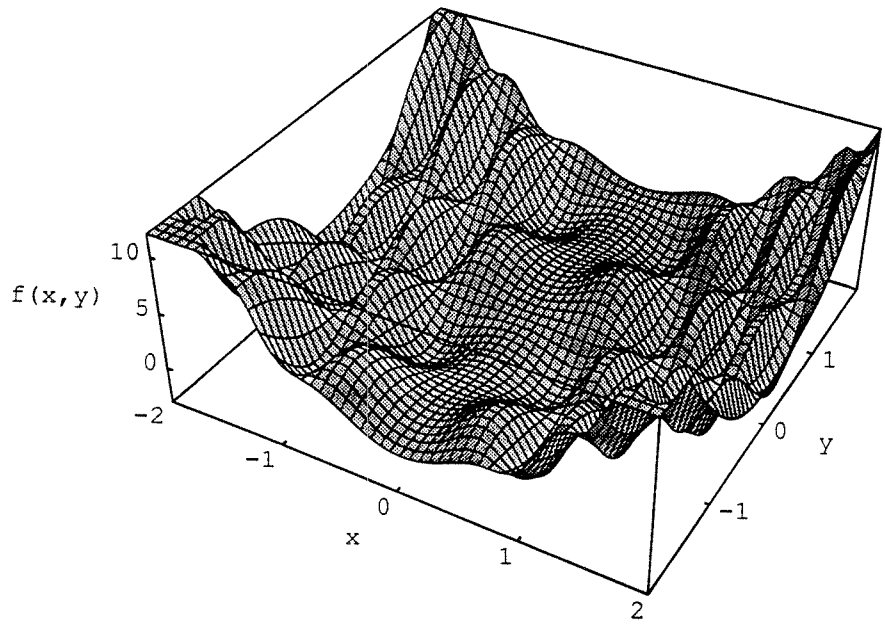

Figure 2.4A: Example of a two-dimensional function.

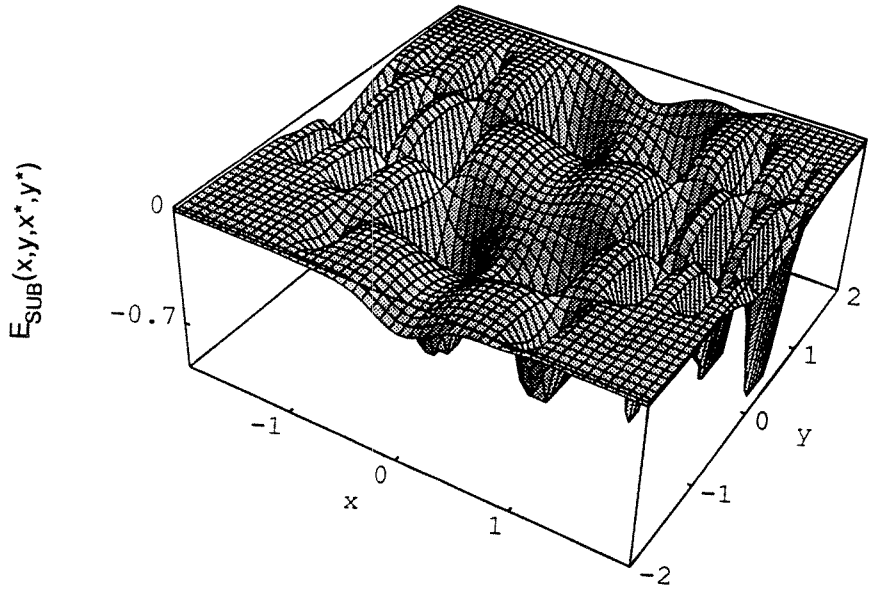

Figure 2.4B: Subenergy tunneling transformation applied to the example of Figure 2.4A.

As can be observed, the subenergy function has the following approximate behavior, which is the key to this optimization algorithm:

$$E_{sub}(\vec{x},\vec{x}^*) \simeq \begin{cases} 0, & \hat{f}(\vec{x}) \geq 0, \text{ i.e., } f(\vec{x}) \geq f(\vec{x}^*), \\ \hat{f}(\vec{x}), & \hat{f}(\vec{x}) < 0, \text{ i.e., } f(\vec{x}) < f(\vec{x}^*), \end{cases} \tag{2.16}$$

$$\frac{\partial E_{sub}(\vec{x},\vec{x}^*)}{\partial \vec{x}} \simeq \begin{cases} 0, & f(\vec{x}) \geq f(\vec{x}^*), \\ \frac{\partial f(\vec{x})}{\partial \vec{x}}, & f(\vec{x}) < f(\vec{x}^*). \end{cases} \tag{2.17}$$

Next we summarize and review the properties of terminal repellers.

2.3.2 Terminal Repellers

An equilibrium point $\vec{x}_{eq}$ of the dynamical system

$$\dot{\vec{x}} = \bar{g}(\vec{x}) \tag{2.18}$$

is termed an attractor (repeller) if no (at least one) eigenvalue of the matrix $\mathcal{M}$,

$$\mathcal{M} = \frac{\partial \bar{g}(\vec{x}_{eq})}{\partial \vec{x}}, \tag{2.19}$$

has a positive real part. Typically, dynamical systems such as (2.18) obey the Lipschitz condition

$$\left| \frac{\partial \bar{g}(\vec{x}_{eq})}{\partial \vec{x}} \right| < \infty, \tag{2.20}$$

which guarantees the existence of a unique solution for each initial condition $\vec{x}(0)$. Theoretically, the system's relaxation time to an attractor and escape time from a repeller is infinite, because the transient solution cannot intersect the corresponding solution to which it tends.

Zak, Barhen, and Toomarian [Zak89, ZB90, BTG90, BZT90b, BZT90a] have used the concept of terminal attractors and repellers in the context of neural network dynamics to obviate the infinite-time solution limitations of regular attractors and repellers. Based on

the violation of the Lipschitz condition at equilibrium points, these points induce singular solutions such that each solution approaches the terminal attractor or escapes from the terminal repeller in finite time.

For example, the system

$$\dot{x} = -x^{1/3} \tag{2.21}$$

has an attracting equilibrium point at $x = 0$ which violates the Lipschitz condition,

$$\left|\frac{d\dot{x}}{dx}\right| = \left|-\frac{1}{3}x^{-2/3}\right| \to \infty, \qquad \text{as} \qquad x \to 0. \tag{2.22}$$

The attractor is termed terminal, since from any initial condition $x_0 \neq 0$, the dynamical system in (2.21) reaches the equilibrium point $x = 0$ in a finite time,

$$t_0 = -\int_{x_0}^{x \to 0} x^{-1/3} dx = (3/2)x_0^{2/3}. \tag{2.23}$$

Similarly, the dynamical system

$$\dot{x} = x^{1/3} \tag{2.24}$$

has a repelling unstable equilibrium point at $x = 0$ which violates the Lipschitz condition. Any initial condition which is infinitesimally close to the repelling point $x = 0$ will escape the repeller, to reach point x_0 in a finite time,

$$t_0 = \int_{\epsilon \to 0}^{x_0} x^{-1/3} dx = (3/2)x_0^{2/3}. \tag{2.25}$$

The behavior of the terminal attractor and repeller is shown in Figure 2.5. Terminal repellers, in conjunction with the subenergy tunneling function introduced above, form the basis of our global optimization algorithm.

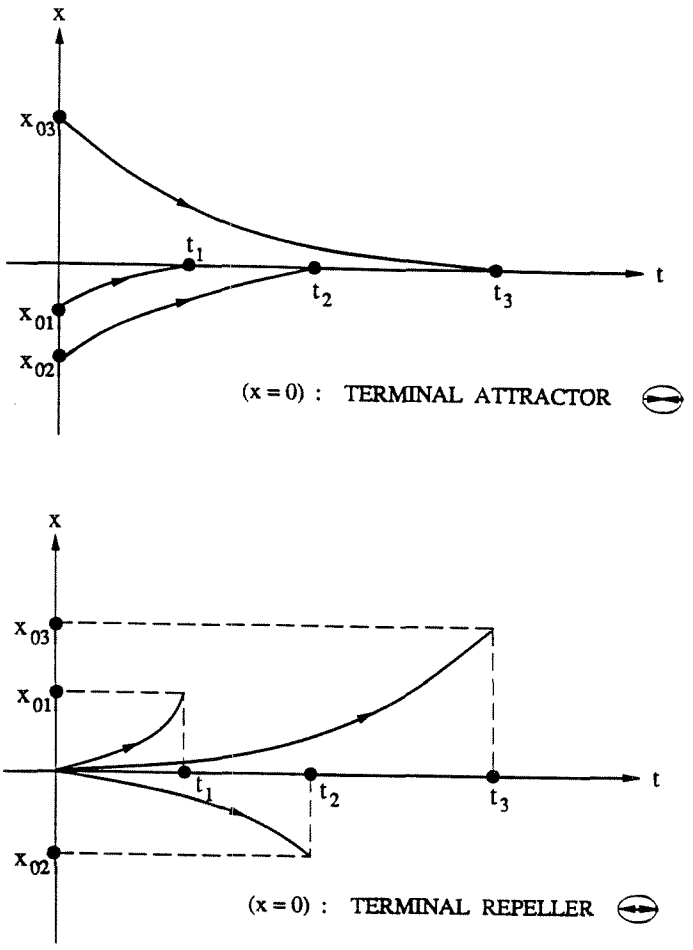

Figure 2.5: Behavior of terminal attractor and repeller.

2.3.3 TRUST Algorithm: One-Dimensional Case

We now assemble the above concepts into the TRUST global optimization scheme. For simplicity, the case of one-dimensional optimization is considered first. Section 2.5 discusses the multi-dimensional case.

Let $f(x)$ be a scalar function which is to be globally minimized over a given interval

$[x_L, x_U]$. Define a new cost function to be minimized,

$$\begin{aligned} E(x, x^*) &= \log\left(\frac{1}{1 + exp(-(\hat{f}(x) + a))}\right) - \frac{3}{4}k(x - x^*)^{4/3}u(\hat{f}(x)) \\ &= E_{sub}(x, x^*) - k\ E_{rep}(x, x^*)u(\hat{f}(x)). \end{aligned} \tag{2.26}$$

The Heaviside step function $u(\cdot)$ was defined in (2.9), and $\hat{f}(x) = f(x) - f(x^*)$, as in (2.13). The first term in the right-hand side of Equation (2.26) corresponds to the subenergy function. The second term is referred to as the repeller energy term, i.e., a term which when differentiated will yield an expression of the form (2.24). The parameter $k > 0$ is referred to as the power of the repeller. The selection of its value will be addressed below.

Application of gradient descent to $E(x, x^*)$ in (2.26) results in the dynamical system

$$\begin{aligned} \dot{x} &= -\frac{\partial E(x, x^*)}{\partial x} \\ &= -\frac{\partial f(x)}{\partial x}\ \frac{1}{1 + exp(\hat{f}(x) + a)}\ +\ k(x - x^*)^{1/3}u(\hat{f}(x)) \\ &+\ \frac{3}{4}k(x - x^*)^{4/3}\delta(\hat{f}(x)). \end{aligned} \tag{2.27}$$

The third term in the r.h.s. of Equation (2.27) is identically zero for any x. Consequently, (2.27) simplifies to

$$\dot{x} = -\frac{\partial f(x)}{\partial x}\ \frac{1}{1 + exp(\hat{f}(x) + a)} + k(x - x^*)^{1/3}u(\hat{f}(x)). \tag{2.28}$$

Equation (2.28) represents gradient descent on $E(x, x^*)$; therefore, its equilibrium state will be a local minimizer of $E(x, x^*)$.

To qualitatively discuss the behavior of this system, we refer to the components of (2.28) as follows:

$$\begin{aligned} \frac{1}{1 + exp(\hat{f}(x) + a)} &= \text{gradient multiplier;} \\ -\frac{\partial f(x)}{\partial x}\ \frac{1}{1 + exp(\hat{f}(x) + a)} &= \text{subenergy gradient;} \end{aligned}$$

$$k(x - x^*)^{1/3} u(\hat{f}(x)) \quad = \quad \text{repeller term.}$$

The dynamical system (2.28) autonomously switches between the following two phases:

Phase I. This phase, which is effectively a tunneling phase, is characterized by $f(x) \geq f(x^*)$. Since the gradient multiplier rapidly tends toward zero for increasing $\hat{f}(x)$, the subenergy gradient magnitude is nearly zero,

$$\frac{\partial E_{sub}(x, x^*)}{\partial x} \simeq 0.$$

In other words, the subenergy function is nearly flat and approximately zero in magnitude for $f(x) \geq f(x^*)$. Since the subenergy gradient magnitude is negligible compared to the magnitude of the repeller term, in this phase (2.28) behaves approximately as

$$\dot{x} \simeq k(x - x^*)^{1/3}.$$

Thus, the dynamical system (2.28) is repelled from x^* across the surface of the flattened subenergy tunneling function, until $f(x) < f(x^*)$. In effect, this phase tunnels through portions of $f(x)$ where $f(x) \geq f(x^*)$.

Phase II. In this phase, which is a minimization phase, $f(x) < f(x^*)$. The gradient multiplier term has approximately unit magnitude, and the repeller term is identically zero. Thus, (2.28) behaves approximately as

$$\dot{x} \simeq -\frac{\partial f(x)}{\partial x}.$$

This phase implements minimization via gradient descent.

In summary, Equation (2.28) behaves approximately as:

$$\dot{x} \simeq \begin{cases} k(x - x^*)^{1/3}, & f(x) \geq f(x^*), \\ -\frac{\partial f(x)}{\partial x}, & f(x) < f(x^*). \end{cases} \tag{2.29}$$

A more detailed analysis of TRUST algorithm represented by (2.28) is considered below.

2.3.4 Initial Conditions and Overview of the TRUST Algorithm Operation

In the one-dimensional case,

$$\mathcal{D} = [x_L \leq x \leq x_U].$$

To initiate optimization, x^* is chosen to be one of the boundary points of $\mathcal{D}$. In effect, a repeller is placed at x^*, and the dynamical system in (2.28) is given initial conditions $x^* + \epsilon$, where ϵ is a small perturbation which drives the system into the domain of interest.

Remark 2.1. Consistency in the flow direction is necessary, i.e., ϵ is of constant sign throughout a particular optimization. A system will be termed "positive flow" if it is initiated at x_L and $\epsilon > 0$ is consistently chosen. Likewise, a system is termed "negative flow" if it is initiated at x_U, and $\epsilon < 0$ is consistently chosen.

The selection of x^* defines a zero subenergy limit $f(x^*)$ above which $E_{sub}(x, x^*)$ is nearly zero in value and approximately flat. If $f(x^* + \epsilon) < f(x^*)$, the system immediately enters a gradient descent phase (Phase II above) which equilibrates at $x = x^{1(*)}$. Typically, $x^{1(*)}$ is a local minimum, though it could be an inflection point (or saddle point in higher dimensions). We refer to $x^{1(*)}$ as a "lower critical point." Here we assume it is a local minimum, though the case of an inflection point is considered in the sequel.

We then set $x^* = x^{1(*)}$ in (2.28), and perturb x to $x^* + \epsilon$. Since $x^{1(*)}$ is a local minimum, $f(x) \geq f(x^*)$ in a neighborhood of x^*. Consequently, the repelling term is active in this phase (Phase I above). Although the gradient of the objective function is uphill, the associated subenergy surface is essentially flat in the vicinity of x^*. If the magnitude of k is chosen sufficiently large (see below), the repeller located at x^* repels the system across the flattened subenergy surface, which in effect pushes the system up the hill of the associated objective function surface. The dynamical system remains in the repelling phase until it reaches a lower basin of attraction, where $\hat{f}(x) < 0$. In effect, this phase tunnels through al

of the state space region with functional values that lie above that of the last found lower critical point $f(x^{1(*)})$.

As the dynamical system enters the next basin, $\hat{f}(x) < 0$, the algorithm automatically switches to gradient descent, leading to minimization of $f(x)$. The system will equilibrate at the next lower local minimum $x^{2(*)}$ (i.e., $f(x^{2(*)}) < f(x^{1(*)})$). We set $x^* = x^{2(*)}$ and repeat the process. This is shown graphically in Figures 2.6A–2.6D.

If $f(x^* + \epsilon) \geq f(x^*)$ when the optimization procedure is initiated, (2.28) is initially in a tunneling phase. The tunneling will proceed to a lower basin, at which point it enters a gradient descent phase and follows the behavior discussed above.

A sufficient value of k to ensure tunneling can be determined as follows. After reaching a local critical point x^*, the zero energy limit is reset, effectively placing a repeller at the minimum x^*. The dynamical system is restarted with initial condition $x_0 = x^* + \epsilon$, where $\epsilon > 0$ (assuming positive flow). The repeller need only be strong enough to push the system over the relatively flattened surface. If x^* is an inflection point, then *any* positive value of k is sufficient. If x^* is a local minimum, then for $\dot{x}$ to be positive when the positive flow dynamical system is restarted at the perturbed location $x_0 = x^* + \epsilon$, the following condition must be satisfied:

$$k(x_0 - x^*)^{1/3} > \frac{\partial f(x_0)}{\partial x} \frac{1}{1 + exp(\hat{f}(x_0) + a)}. \tag{2.30}$$

A sufficient condition to satisfy (2.30) is that

$$\begin{aligned} k &> \left(\frac{1}{\epsilon^{1/3}(1 + exp(\hat{f}(x_0) + a))} \right) \frac{\partial f(x_0)}{\partial x} \\ &\simeq \left(\frac{\epsilon^{2/3}}{1 + exp(a)} \right) \frac{\partial^2 f(x^*)}{\partial x^2}. \end{aligned} \tag{2.31}$$

Note that ϵ is typically a small number, like 0.001 or 0.01, hence necessary values of k are typically very reasonable. Thus, stiffness considerations in the integration of (2.28) do not arise from the choice of k. For example, for $\epsilon = 0.01$ and $a = 2$, a value of k such that $k > 0.0056 \frac{\partial^2 f(x^*)}{\partial x^2}$ is sufficiently large to ensure proper tunneling behavior.

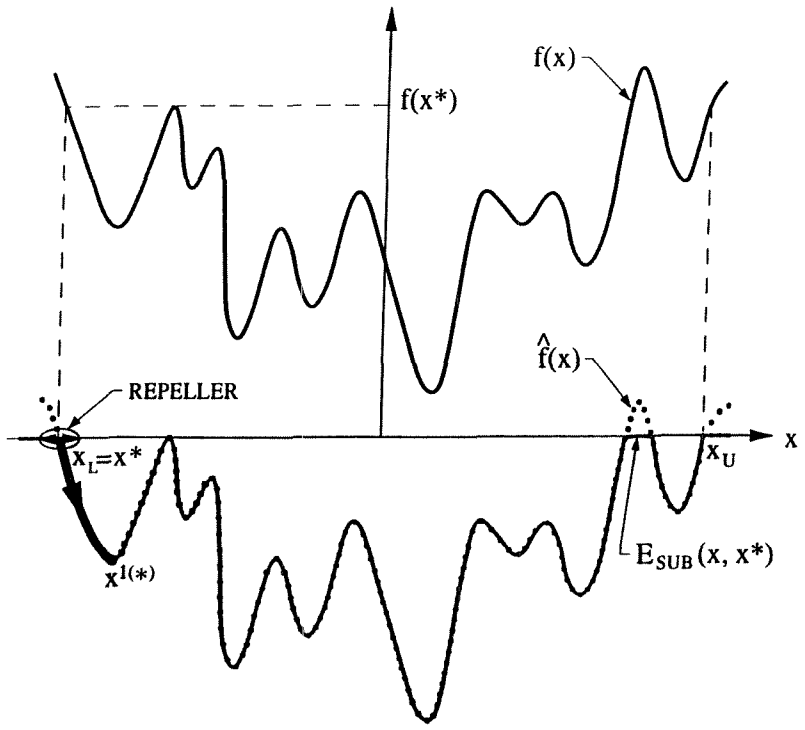

Figure 2.6A: Schematic of TRUST operation (Cycle I).

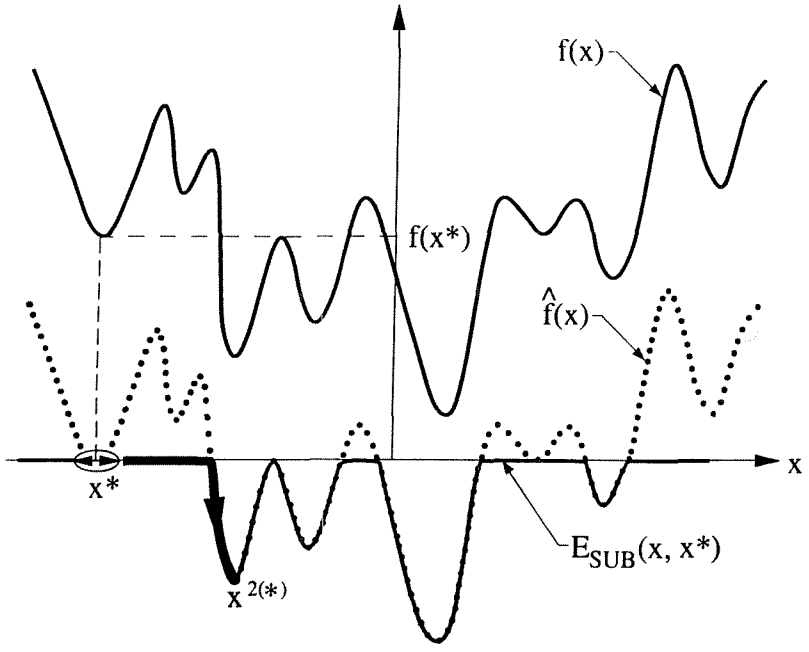

Figure 2.6B: Schematic of TRUST operation (Cycle II).

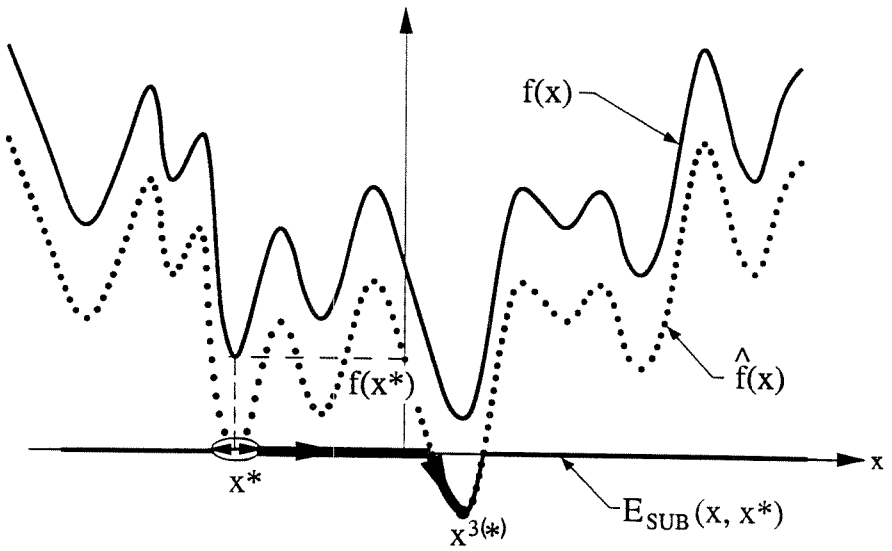

Figure 2.6C: Schematic of TRUST operation (Cycle III).

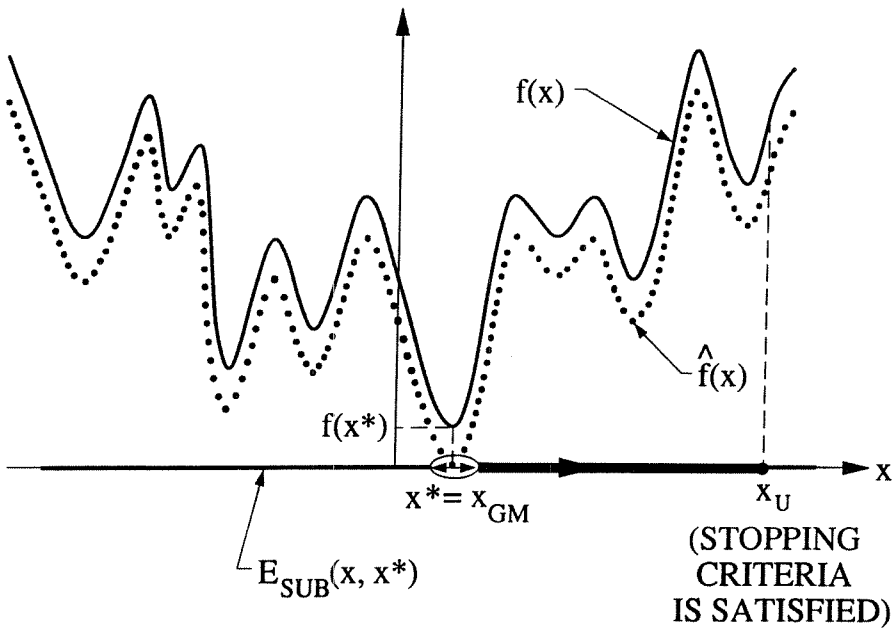

Figure 2.6D: Schematic of TRUST operation (Cycle IV).

During the remainder of the tunneling phase, we need only ensure that at any point x',

$$k > \frac{1}{(x' - x^*)^{1/3}} \frac{\partial f(x')}{\partial x} \left(\frac{1}{1 + exp(\hat{f}(x') + a)} \right). \tag{2.32}$$

Note that the value of k computed using (2.31) at the beginning of the tunneling process is almost always sufficiently large for the entire tunneling process. The gradient multiplier term decreases at an exponential rate with respect to increasing $\hat{f}(x)$. Thus, $f(x)$ must increase at a rate faster than exponential to ever require an increase in the value of k over the value computed at x_0 in (2.31), i.e., generally (2.31) is sufficient for (2.32). A similar analysis of the negative flow case shows that (2.31) and (2.32) hold in this case as well.

TRUST's implementation of tunneling as a repeller-induced flow over a subenergy surface has a number of advantages over other tunneling methods. First, the tunneling operation is algorithmically and computationally quite simple. Second, if $f(x^{2(*)}) = f(x^{1(*)})$, the associated subenergy surface is still flat, and the system tunnels past this local minimum (or inflection point) into a basin with a lower local minimum. This feature eliminates the difficulty with multiple poles in the tunneling algorithm of Levy and Montalvo. Third, convergence of the gradient descent phase to an inflection point does not cause a problem, as the dynamical system will escape the inflection point during the next gradient descent phase.

It must be stressed that TRUST was developed to be implemented in continuous analog circuitry, where the integration of (2.28) is stable. In digital computer implementations, some care must be exercised during the numerical integration of (2.28) to ensure that a basin of attraction is not jumped over due to the finite-step-length integration of (2.28). Determination of an appropriate step size could follow from [GS90]. Finally, the TRUST tunneling method will always reach a point in the adjacent basin of attraction with lower functional values. Other tunneling methods which find zeros of a tunneling function are not guaranteed to find the most adjacent tunneling point, and therefore have complicated and less reliable stopping criteria. The TRUST stopping criterion is outlined below, and a more detailed examination of the convergence behavior of TRUST is given in Section 2.4.

2.3.5 Stopping Criteria

The successive minimization and tunneling computational processes continue until a suitable stopping criterion is satisfied. For the one-dimensional case, the stopping criterion is quite simple. As soon as a local minimum x_{LM} in $\mathcal{D}$ has been reached, the optimization cycle is repeated by placing a repeller at x_{LM} and perturbing the system to initiate the next tunneling phase. If x_{LM} were the lowest local minimum (i.e., if $x_{LM} = x_{GM}$), the subenergy transformation would flatten $f(x)$ in the entire domain of interest, since $f(x_{GM})$ should be the lowest objective function value in $\mathcal{D}$. The perturbed dynamical system, which is now in a repeller tunneling phase, will eventually flow beyond the upper boundary of $\mathcal{D}$. Assuming positive flow, when the state flows out of the domain boundary, $x > x_U$, the last local minimum found is taken as the global minimum. Recall that we assume that the global minimum does not lie on the boundary of $\mathcal{D}$.

2.4 Analysis of One-Dimensional Convergence

We now examine the convergence of the TRUST algorithm in light of the above discussion. In the one-dimensional case, we seek to globally minimize $f(x)$, a twice differentiable function, over the domain $\mathcal{D} = [x_L, x_U]$. To show that TRUST will converge, under the assumptions of Section 2.1, to a global minimum (if one exists), we analyze its behavior during the different phases of operation. The analysis proceeds as follows. First, the tunneling behavior of TRUST is considered, assuming a local minimum has been found (after an initialization phase). We show that, from a local minimum, the tunneling phase of TRUST reaches a point of the same functional value in an adjacent basin of attraction of a lower critical point, or flows to a boundary of $\mathcal{D}$ if no such point exists. Next, we show the obvious result that the gradient descent behavior of TRUST will converge to a lower critical point. An inductive analysis of these two phases leads to the global minimization behavior and stopping criterion. Finally, we consider the initialization of the TRUST algorithm, showing that, from all possible initial conditions, TRUST will reach the first effective local minimum,

if it exists, or flow out of $\mathcal{D}$. The case of inflection points are considered throughout the discussion as necessary.

Let us first consider the tunneling behavior of TRUST after finding the jth local minimum $x^{j(*)}$ of $f(x)$. To simplify the discussion, introduce the following notation. Let

$$\mathcal{D}_L(x^{j(*)}) = [x_L, x^{j(*)}) \quad \text{and} \quad \mathcal{D}_U(x^{j(*)}) = (x^{j(*)}, x_U]$$

respectively be termed the lower and upper domains of $x^{j(*)}$. Let $S_L(x^{j(*)})$ and $S_U(x^{j(*)})$ respectively denote the sets of lower and upper tunneling points of $x^{j(*)}$,

$$S_L(x^{j(*)}) = \{x \in \mathcal{D}_L \mid f(x) = f(x^{j(*)}) \}, \tag{2.33a}$$

$$S_U(x^{j(*)}) = \{x \in \mathcal{D}_U \mid f(x) = f(x^{j(*)}) \}. \tag{2.33b}$$

That is, $S_L(x^{j(*)})$ and $S_U(x^{j(*)})$ are points in $\mathcal{D}$ with the same functional values as $f(x^{j(*)})$. Note that $S_L(x^{j(*)})$, or $S_U(x^{j(*)})$, or possibly both, may be empty sets depending upon $x^{j(*)}$, $f(x)$ and the chosen direction of flow. If $S_L(x^{j(*)})$ or $S_U(x^{j(*)})$ are not empty, define the adjacent lower and upper tunneling points as follows:

$$x_{A_L} = \min_{x \in S_L(x^{j(*)})} ||x - x^{j(*)}||, \tag{2.34a}$$

$$x_{A_U} = \min_{x \in S_U(x^{j(*)})} ||x - x^{j(*)}||. \tag{2.34b}$$

If either $S_L(x^{j(*)})$ or $S_U(x^{j(*)})$ are empty, define the adjacent tunneling points respectively as

$$x_{A_L} = x_L, \tag{2.35a}$$

$$x_{A_U} = x_U. \tag{2.35b}$$

Now define the lower and upper tunneling intervals:

$$\mathcal{D}_{T_L}(x^{j(*)}) = [x_{A_L}, x^{j(*)}), \tag{2.36a}$$

$$\mathcal{D}_{T_U}(x^{j(*)}) = (x^{j(*)}, x_{A_U}]. \tag{2.36b}$$

Finally, we construct the tunneling interval $\mathcal{D}_T(x^{j(*)})$ as follows:

$$\begin{aligned} \mathcal{D}_T(x^{j(*)}) &= \mathcal{D}_{T_L}(x^{j(*)}) \cup \mathcal{D}_{T_U}(x^{j(*)}) \cup \{x^{j(*)}\} \\ &= \{x \in \mathcal{D}(x) \mid x_{A_L} \leq x \leq x_{A_U}\}. \end{aligned} \tag{2.37}$$

That is, the tunneling region, $\mathcal{D}_T(x^{j(*)})$, is the connected interval containing $x^{j(*)}$ and whose endpoints are either points with the same functional value (and thus points for initiating a subsequent local optimization phase) or a boundary of $\mathcal{D}$. Note, that $f(x)$ may assume local minima, maxima, and inflection points in $\mathcal{D}_T(x^{j(*)})$, though

$$f(x) \geq f(x^{j(*)}), \qquad \forall x \in \mathcal{D}_T(x^{j(*)}).$$

We wish to show that the dynamical system (2.28) is unstable on $\mathcal{D}_T(x^{j(*)})$ and will flow toward the boundary of this interval (thus performing the tunneling operation, or satisfying the stopping criterion). To do this, we define a Lyapunov energy function

$$\widetilde{E}(x(t), x^{j(*)}) = \frac{3}{4}k(x(t) - x^{j(*)})^{4/3}. \tag{2.38}$$

We note that $\widetilde{E}(x)$ is positive definite on $\mathcal{D}_{T_L}(x^{j(*)})$ and $\mathcal{D}_{T_U}(x^{j(*)})$, and is positive semi-definite in $\mathcal{D}_T(x^{j(*)})$, assuming a zero value only at $x^{j(*)}$. Further, note that $\widetilde{E}(x, x^{j(*)})$ is a strictly increasing function of $||x - x^{j(*)}||$ on both $\mathcal{D}_{T_L}(x^{j(*)})$ and $\mathcal{D}_{T_U}(x^{j(*)})$, and respectively assumes its maximum values on the lower and upper boundaries of $\mathcal{D}_{T_L}(x^{j(*)})$ and $\mathcal{D}_{T_U}(x^{j(*)})$. The time derivative of $\widetilde{E}(x(t), x^{j(*)})$ is

$$\frac{d}{dt}\widetilde{E}(x, x^{j(*)}) = k(x - x^{j(*)})^{1/3}\dot{x}$$

$$= k^2(x - x^{j(*)})^{2/3} - k\frac{\partial f(x)}{\partial x}\left(\frac{1}{1 + exp(\hat{f}(x) + a)}\right)(x - x^{j(*)})^{\frac{1}{3}}. \quad (2.39)$$

Recall that k is a positive constant. From the discussion in Subsection 2.3.4, also recall that the value of k for positive flow is chosen so that $\dot{x} > 0$ on $\mathcal{D}_{T_U}(x^{j(*)})$. Similarly, for negative flow, k is chosen so that $\dot{x} < 0$ on $\mathcal{D}_{T_L}(x^{j(*)})$. Note that for both cases (i.e., positive and negative flow), the same sufficient condition for k in (2.31) holds. Thus, $(d/dt)\widetilde{E}(x, x^{j(*)})$ is positive on $\mathcal{D}_{T_L}(x^{j(*)})$ and $\mathcal{D}_{T_U}(x^{j(*)})$; $(d/dt)\widetilde{E}(x, x^{j(*)})$ assumes a zero value only at $x^{j(*)}$. This implies that $||x - x^{j(*)}||$ must also be increasing with time on $\mathcal{D}_{T_L}(x^{j(*)})$ or $\mathcal{D}_{T_U}(x^{j(*)})$. That is, from any initial condition in $\mathcal{D}_{T_L}(x^{j(*)})$, (2.28) will flow to x_{A_L} (negative flow). Similarly, from any initial condition in $\mathcal{D}_{T_U}(x^{j(*)})$, (2.28) will flow to x_{A_U} (positive flow).

Hence, we have just shown that (2.28), when perturbed to $x^{j(*)} + \epsilon$, will flow to a point $x^{j(0)}$ whose functional value is just below $f(x^{j(*)})$, i.e., $f(x^{j(0)}) \stackrel{\sim}{<} f(x^{j(*)})$. If no such point exists, the system will flow to the boundary of $\mathcal{D}$. Also note that the analysis shows that:

1) Any nonzero perturbation size ϵ leads to correct tunneling behavior.

2) Because of the properties of the terminal repellers, the tunneling flow must occur in finite time.

We also need to consider the behavior of the tunneling phase if $x^{j(*)}$ is actually an inflection point, and not a local minimum. Assume that the inflection point $x^{j(*)}$ was reached by a minimization phase which originated in $\mathcal{D}_L$ (i.e., from a positive flow system). In this case, $\mathcal{D}_{T_U}(x^{j(*)})$ is a zero length interval. A small perturbation $x^{j(*)} + \epsilon$ will put TRUST in another gradient descent phase. Similarly, if $f(x)$ is infinitely degenerate, and thus flat in $\mathcal{D}_U(x^{j(*)})$, the repeller induced flow will push the system over the degenerate interval.

Next, consider the behavior of the TRUST dynamical system in a gradient descent phase. Assume that a tunneling phase has been completed, and we are at point $x^{j(0)}$. This point must be within a basin of attraction of a lower local critical point, such that

$|\partial f(x)/\partial x| \neq 0$, and $f(x^{j(0)}) \tilde{<} f(x^{j(*)})$ holds. The dynamical system (2.28) then becomes

$$\dot{x} = -\frac{\partial f}{\partial x}\left(\frac{1}{1+exp(f(x)-f(x^{j(*)})+a)}\right). \tag{2.40}$$

Again, we can analyze the convergence properties of this system by defining a Lyapunov energy function,

$$\widehat{E}(x) = f(x) - f(x^{j+1(*)}),$$

where $x^{j+1(*)}$ is the next adjacent lower critical point of $f(x)$ and $\widehat{E}(x)$ is defined on the interval $[x^{j(0)}, x^{j+1(*)}]$.

The time derivative of $\widehat{E}(x)$ in the domain is

$$(d/dt)\widehat{E}(x) = \frac{\partial f(x)}{\partial x}\dot{x} = -\left(\frac{\partial f}{\partial x}\right)^2 \frac{1}{1+exp(\hat{f}(x)+a)}, \tag{2.41}$$

which is a negative semidefinite function, assuming zero value only at $\frac{\partial f(x)}{\partial x} = 0$. Thus, from $x^{j(0)}$, the dynamical system (2.28) will converge to a lower critical point $x^{j+1(*)}$, where we reset x^* to $x^{j+1(*)}$ and repeat the same process, and the above analysis procedure holds.

Thus, the above analysis has shown that, starting from a local minimum or inflection point $x^{j(*)}$, and applying the algorithm outlined in Section 2.3, Equation (2.28) will converge to another local minimum (or inflection point) $x^{j+1(*)}$ with $f(x^{j+1(*)}) < f(x^{j(*)})$, or flow out of $\mathcal{D}$ if there are no lower minima. We call $x^{j+1(*)}$ the *next effective local minimum*, as there may be many local minima located between $x^{j(*)}$ and $x^{j+1(*)}$, but these lower minima have functional values greater than $f(x^{j(*)})$. Thus, by the inductive analysis of the two above phases, TRUST (assuming a positive flow system) will find a sequence of effective minima,

$$x^{1(*)} < x^{2(*)} < \cdots < x^{l(*)}, \tag{2.42}$$

such that "global descent property" is implemented as follows:

$$f(x^{1(*)}) > f(x^{2(*)}) > \cdots > f(x^{l(*)}). \tag{2.43}$$

From $x^{l(*)}$, (2.28) will flow to x_U, and we know from the above discussion that no lower local minima can exist in the interval $(x^{l(*)}, x_U]$. Thus, the last local minimum found must be the global minimum.

The above inductive analysis assumed that the TRUST algorithm was initiated at local minimum $x^{1(*)}$. We now turn to the operation of TRUST from its initial conditions, to show that it will converge to the first effective local minimum $x^{1(*)}$, if it exists. From there, the previous inductive analysis holds. Assume a positive flow system (a similar analysis holds for negative flow). Several possible different initial conditions at x_L have to be considered.

Case 1. x_L **is a local minimum.** The above analysis holds immediately.

Case 2. x_L **is an inflection point.** If $f(x)$ is increasing in a positive flow neighborhood of x_L, then an upper tunneling region exists. Initiation of (2.28) at $x_L + \epsilon$ will initiate a tunneling phase, which as shown above will either flow out of the domain $\mathcal{D}$ if no global minimum (that satisfies the local minima constraints (2.3) and (2.4)) exists, or will reach a point where subsequent gradient descent converges to the first effective local minimum. If $f(x)$ is decreasing in a positive flow neighborhood of x_L, then the system enters a gradient descent phase, which will converge to a lower local critical point.

Case 3. x_L **is a local maximum.** Initiating (2.28) at $x_L + \epsilon$ puts (2.28) in a gradient descent phase, which will converge to $x^{1(*)}$.

Case 4. $\partial f(x)/\partial x > 0$ **at** x_L**.** An upper tunneling region $D_{T_U}(x_L)$ exists. According to the previous analysis, perturbing x to $x_L + \epsilon$ will cause (2.28) to reach either an adjacent tunneling point, where subsequent gradient descent will find the first effective local minimum (or inflection point) $x^{1(*)}$, or will flow to x_U if in fact $f(x_L)$ is the lowest value $f(x)$ assumes in $\mathcal{D}$.

Case 5. $\partial f(x)/\partial x < 0$ **at** x_L**.** At $x_L + \epsilon$ (2.28) immediately enters a gradient descent phase, converging to the first effective local minimum (or inflection point), if one exists; else, gradient descent will flow to x_U if no such point exists in $\mathcal{D}$.

Thus, in the continuous case and under the assumptions in Section 2.1, TRUST is guaranteed to find the global minimum in a one-dimensional interval. If the function is degenerate (i.e., several global minima), TRUST will determine only the first encountered global minimum. In order to locate the consequent global minima, we iteratively reset x_L to $x_{GM} + \epsilon$ and restart there.

2.5 TRUST Algorithm: Multi-Dimensional Case

The one-dimensional algorithm of Section 2.3 can be extended to handle multi-dimensional global optimization, though convergence to the global minimum is not absolutely guaranteed. Let $f(\vec{x}\,)$ be a function of the $n \times 1$ state vector $\vec{x}$, and define the multi-dimensional functional

$$\begin{aligned} E(\vec{x},\vec{x}^{\,*}) &= log\left(\frac{1}{1+exp(-(\hat{f}(\vec{x}\,)+a))}\right) - k\frac{3}{4}\sum_{j=1}^{n}(x_j - x_j^*)^{4/3}u(\hat{f}(\vec{x}\,)) \\ &= E_{sub}(\vec{x},\vec{x}^{\,*}) - k\ E_{rep}(\vec{x},\vec{x}^{\,*})\ u(\hat{f}(\vec{x}\,)). \end{aligned} \tag{2.44}$$

The multi-dimensional subenergy term is analogous to the one-dimensional subenergy function. The portions of the objective function surface which lie above the zero subenergy limit $f(\vec{x}^*)$ are flattened by the use of the subenergy function (as shown in Figure 2.4B).

Upon application of gradient descent to $E(\vec{x},\vec{x}^{\,*})$ in (2.44), we obtain the dynamical system

$$\dot{x}_j = -\frac{\partial f(\vec{x})}{\partial x_j}\left(\frac{1}{1+exp(\hat{f}(\vec{x}\,)+a)}\right) + k(x_j - x_j^*)^{1/3}u(\hat{f}(\vec{x}\,)), \tag{2.45}$$

where x_j denotes the jth component of $\vec{x}$. Equation (2.45) has a highly parallel structure consisting of n weakly coupled differential equations. This dynamical system is analogous to the dynamical system described by Equation (2.28). The initial conditions, operation, and stopping criterion for Equation (2.45) are also highly analogous to those discussed above.

In the multi-dimensional case, $\vec{x}^*$ is initially chosen to be one corner of the hyper-parallelpiped $\mathcal{D}$, usually $x_i^* = x_{iL},\ \forall i$. A repeller is placed at $\vec{x}^*$. It should be noted that

the repelling terms in the multi-dimensional case can be interpreted as hyperplane repellers, and are active whenever $\hat{f}(\vec{x}) \geq 0$. The initial state of the system is set to $\vec{x}^* + \vec{\epsilon}$, where $\vec{\epsilon}$ is a small perturbation which drives the system into $\mathcal{D}$. We assume $\vec{\epsilon}$ has uniform direction during the optimization, analogous to the consistent positive or negative flow operation of the one-dimensional algorithm. Depending upon the relative values of $f(\vec{x}^*)$ and $f(\vec{x}^* + \vec{\epsilon})$, the dynamical system will initially be in a tunneling phase or a gradient descent phase. These phases are analogous to the one-dimensional case. An appropriate value for the repeller power k can be determined by analogy to (2.30)–(2.32). The multi-dimensional stopping criterion is also similar to the one-dimensional case. When the system state flows out of the domain boundaries, the last local minimum found is taken as the global minimum.

Theoretically, convergence of the method to a global minimum is not formally guaranteed in the multi-dimensional case due to the constant perturbation direction vector $\vec{\epsilon}$. However, in practice, as a result of its global descent property, the system dynamics escapes local minima valleys with help of the repeller effect, and flows into lower valleys of the objective function using the information it gets from the gradient term.

2.6 Benchmarks and Comparison to Other Methods

This section presents results of benchmarking tests carried out for the TRUST algorithm using several standard one- and multi-dimensional test functions taken from the literature. In Tables 2.1–2.4 the performance of TRUST is compared to well-known global optimization procedures. Specifically in Tables 2.3 and 2.4, TRUST is compared against the best competing global optimization methods, where the term "best" indicates the best widely reported results the authors could find for the particular benchmark test function. The criteria for comparison is the number of function evaluations. For the TRUST algorithm, the function evaluation count includes every iteration from the initial conditions to the satisfaction of the stopping criterion outlined in Subsection 2.3.5. We note that in *every* benchmark, TRUST converged to the global minimum.

In accordance with Subsection 2.3.1 the constant a assumes the value $a = 2$ in the sequel.

Furthermore, Equation (2.28) was integrated using a simple Euler integration scheme; that is,

$$\tau \dot{\vec{x}} = \tau \frac{\vec{x}(k+1) - \vec{x}(k)}{\Delta t} = -\frac{\partial E(\vec{x}, \vec{x}^*)}{\partial \vec{x}}, \tag{2.46}$$

where Δt is the step size. The time constant τ is taken to be 1 in all cases studied here. For highly nonlinear and stiff objective functions, more robust integration schemes are preferable ([ZB90] and [BTG90]). We note that for Euler integration, the selection of the integration stepsize must be done carefully to ensure stability. We do not provide an analysis of the step size in this thesis, since (as we have previously stated) our ultimate goal was implementation of this algorithm in continuous analog VLSI circuitry (Chapter 4), where such considerations do not apply.

Table 2.1: Comparison of TRUST and other algorithms based on number of function evaluations.

	Method					
Function	SM	TM	DT	IM	FFA	TRUST
1(i)	10822	1496	1469			168
1(ii)	10822	1496	1132			168
1(iii)	10822	1496				32
1(iv)					375	76
2(i)	241215	12160	6000	7424		588
2(ii)	241215	12160	6000	7424		269
2(iii)					408	256

A description of each test function, the relevant initial conditions, domain of interest ($\mathcal{D}$), TRUST parameters, and integration step size are given in the Appendix A. It should be noted that in each comparison, the relevant parameters (such as initial condition) are the same as those used in the reported literature. In Tables 2.1 and 2.2 the following abbreviations are used: SM is the stochastic method [APPZ85]; TM is the tunneling method [LM85]; DT is the dynamic tunneling method presented in [Yao89]; IM is the interval method [WHS85]; FFA is the filled function approach [Ge90]; and FSA is the fast simulated annealing method [SH87].

Table 2.2: Comparison of TRUST and other algorithms based on number of function evaluations.

	Method			
Function	SM	DT	FSA	TRUST
3(i)				38
3(ii)		1414		22
3(iii)				21
3(iv)		7871	9228	21
4(i)	19940			74
4(ii)				58
5(i)	7390			40
5(ii)	4853			94
5(iii)	8235			163
5(iv)	27859			1449

Table 2.3: Comparison of TRUST and other algorithms based on number of function evaluations.

	Method						
Function	SA	MRS	P	CRS	SCA	MLSL	TRUST
6	5917	1176	179				77
7		160	133	1800	1558	206	60

In Table 2.3, SA is an abbreviation of simulated annealing [KGV83]; MRS is the multiple random start method [Bre70]; P is an abbreviation of the P-algorithm [TZ89]; CRS is the controlled random search of Price [Pri78]; SCA is the search clustering approach of Törn [Tor78]; and MLSL is the multi-level single linkage method [KT85].

In Table 2.4, PIJ, BAT, STR, ZIL, and BRE are respectively abbreviations for the results of Pijavskij, Batishchev, Strongin, Zilinskas, and Brent [TZ89].

Table 2.4: Comparison of TRUST and other algorithms based on number of function evaluations.

	Method					
Function	PIJ	BAT	STR	ZIL	BRE	TRUST
8	462	120	45	33	25	19
9(i)	3817	816	150	125	161	69
9(ii)	3817	816	150	125	161	99

2.7 Summary

We have introduced TRUST, a novel deterministic methodology for unconstrained global function optimization, which combines the concept of terminal repellers with a new subenergy tunneling function. Global optimization is formulated as the solution to a system of deterministic differential equations which incorporate these novel features. The flow of this dynamical system leads to global optimization. It was shown that, under very general assumptions [see Section 2.1], the algorithm is provably convergent to the global minimum in the one-dimensional case. In higher dimensions it exhibits a global descent property.

Benchmark comparisons (Section 2.6) with other global optimization procedures have demonstrated that TRUST is significantly faster, as measured by the number of function evaluations, than the best currently available methods for these standard functions. Furthermore, our algorithm systematically converged to the global minimum in all benchmark simulations, even in the multi-dimensional case. However, one can construct hypothetical functions for which TRUST may not find the global minimum, depending upon the initial conditions.

This chapter considered TRUST in the context of general objective function optimization. The next chapter discusses the application of TRUST to artificial neural networks, in order to overcome the limitations which arise from local minima during standard training procedures, such as backpropagation.

Chapter 3

Global Optimization in Artificial Neural Networks

3.1 Local Minima Problem Associated with Backpropagation

Backpropagation is a learning algorithm that gives a prescription for changing the connection weights of a feed-forward network to learn a training set of input-output pairs without any prior knowledge of the mathematical function that maps them. More specifically, it uses gradient descent to adjust the weights following the local slope of an error surface towards a minimum. In order to establish the notation, and for the sake of completeness and clarity, we first briefly review the Backpropagation algorithm below.

Without loss of generality, we assume a two-layered feed-forward network with one hidden layer, as shown in Figure 3.1. Let $\vec{\xi}^{\mu}$ and $\vec{t}^{\mu}$, $\mu = 1, 2, \ldots, p$ be the input and target output patterns respectively.

Given pattern μ, hidden unit j receives an input h_j^{μ} and produces an output y_j^{μ},

$$h_j^{\mu} = \sum_k w_{jk}\, \xi_k^{\mu} \quad ; \quad y_j^{\mu} = s(h_j^{\mu}) \tag{3.1}$$

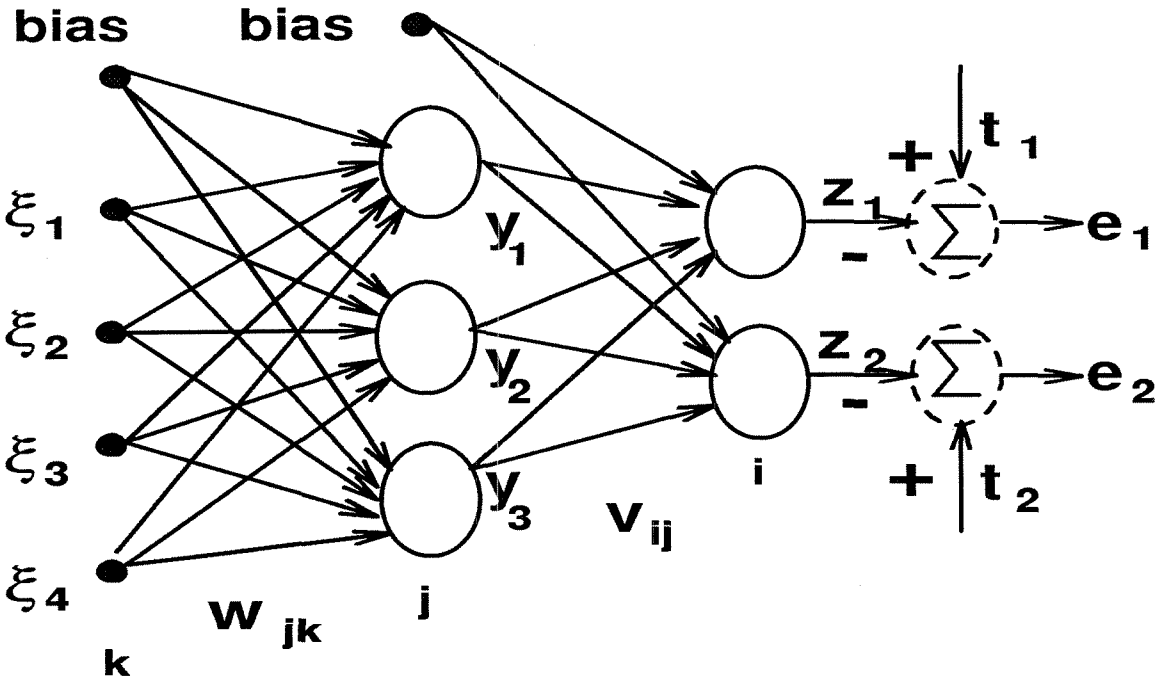

Figure 3.1: A two layer feed-forward network, showing the notation.

where, $s(\cdot)$ is a sigmoidal function of the form $1/(1+exp(-(\cdot)))$. Output unit i receives g_i^μ and produces the final output z_i^μ,

$$g_i^\mu = \sum_j v_{ij}\, y_j^\mu \quad ; \quad z_i^\mu = s(g_i^\mu)\,. \tag{3.2}$$

Bias to the units are not explicitly formulated but can be considered as an extra input clamped to $+1$ and connected to all units in the network.

As an error energy measure, or cost function, we choose the squares of the differences between the actual and target output values summed over the output units and all pairs of

input/output patterns

$$E[\vec{w},\vec{v}] = \frac{1}{2}\sum_{\mu}^{p}\sum_{i}^{I}(t_i^\mu - z_i^\mu)^2 \;=\; \frac{1}{2}\sum_{\mu}^{p}\sum_{i}^{I}(e_i^\mu)^2 \tag{3.3}$$

where, $\vec{w}$ and $\vec{v}$ are the row-concatenated vector representation of the weight matrices $\mathcal{W}$ and $\mathcal{V}$ respectively. To train the network, the weights of each unit are adjusted to minimize the above energy function, thereby reducing the error between the actual and target outputs. In the Backpropagation algorithm, this is accomplished by using gradient descent, which changes the weights in proportion to the negative energy gradient. Thus, it is hoped to find a global minimum to (3.3), which will correspond to the optimal weights that solve the specific mapping problem.

To apply gradient descent, we first calculate the energy gradient for the hidden-to-output connections

$$\frac{\partial E[\vec{w},\vec{v}]}{\partial v_{ij}} = -\sum_{\mu}^{p} e_i^\mu \; s'(g_i^\mu)\; y_j^\mu \tag{3.4}$$

and then for the input-to-hidden connections

$$\frac{\partial E[\vec{w},\vec{v}]}{\partial w_{jk}} = -\sum_{\mu}^{p}\sum_{i}^{I} e_i^\mu \; s'(g_i^\mu)\; v_{ij}\; s'(h_j^\mu)\; \xi_k^\mu \; . \tag{3.5}$$

Finally, we apply gradient descent to equations (3.4) and (3.5) to get the weight dynamics

$$\dot{v}_{ij} = -\eta\,\frac{\partial E[\vec{w},\vec{v}]}{\partial v_{ij}} \quad ; \quad \dot{w}_{jk} = -\eta\,\frac{\partial E[\vec{w},\vec{v}]}{\partial w_{jk}} \tag{3.6}$$

where η is the learning rate.

Thus, training of the network consists mainly of weight adjustments performed according to (3.6). Clearly, gradient descent is a dynamical system whose stable equilibrium point only locally minimizes the error energy function. This works well with simple convex error surfaces, which have a unique minimum, but it often leads to nonoptimal and unacceptable solutions with the highly convoluted nonconvex surfaces encountered in practical problems.

Once the algorithm gets trapped in a local minimum, application of more training iterations fails to improve learning. Ideally, only the global minimum of the error energy in equation (3.3) satisfies the convergence of the output patterns to the desired ones.

In the next section, we propose the "Global Descent" dynamical system, which is based on the TRUST algorithm (Chapter 2), and can be substituted for gradient descent in the Backpropagation algorithm to overcome the aforementioned limitations.

3.2 Global Descent Formalism

The TRUST algorithm has been discussed in Chapter 2 in the context of general unconstrained global function optimization. The algorithm has already been employed, with encouraging results, to robotics applications [BCB91, CB92]. Since the primary purpose of this chapter is the *adaptation of the TRUST formalism to Backpropagation learning* (hereafter named Global Descent), we review here its dynamics from a different perspective; that is, in the context of optimization in artificial neural networks.

Global Descent formulates global optimization as the solution to a system of deterministic differential equations, where $E[\vec{w}, \vec{v}]$ of (3.3) is the function to be optimized with the connection weights being the states of the system. Let $\vec{\varphi}$ denote $(\vec{w}, \vec{v})$ and have elements φ_{ab}. Global Descent is based on the multi-dimensional TRUST dynamics of Equation (2.45), which becomes

$$\dot{\varphi}_{ab} = -\eta \frac{\partial E[\vec{\varphi}]}{\partial \varphi_{ab}} \frac{1}{1 + \exp(E[\vec{\varphi}] - E[\vec{\varphi}^*] + \sigma)} + \eta\, k\, (\varphi_{ab} - \varphi^*_{ab})^{1/3} u(E[\vec{\varphi}] - E[\vec{\varphi}^*]) \quad (3.7)$$

where, $\vec{\varphi}^*$ is a fixed value of $\vec{\varphi}$, which can be a local minimum or an initial weight state, $u(.)$ is the Heaviside step function, and σ is a shifting parameter (its influence is discussed in Subsection 2.3.1, in numerical applications we typically take $\sigma = 2$). The first term in the r.h.s. of equation (3.7) is the subenergy gradient, with $1/(1 + \exp(E[\vec{\varphi}] - E[\vec{\varphi}^*] + \sigma))$ being the gradient multiplier, while the second term is the non-Lipschitzian terminal repeller (Subsection 2.3.2). The parameter $k > 0$ is referred to as the "power" of the repeller, whose

magnitude can be determined using the analysis of Subsection 2.3.4. In the simulations of Section 3.3, $k = 0.001 - 0.01$ gave good results.

The dynamics in (3.7) is achieved upon application of gradient descent to the "cost function"

$$C[\vec{\varphi}, \vec{\varphi}^*] = \log\left(\frac{1}{1+\exp(-(E[\vec{\varphi}] - E[\vec{\varphi}^*] + \sigma))}\right) - k\frac{3}{4}\sum_{ab}(\varphi_{ab} - \varphi^*_{ab})^{4/3} u(E[\vec{\varphi}] - E[\vec{\varphi}^*]), \tag{3.8}$$

which is of the form given in (2.44). Recall that the first term in the r.h.s. of equation (3.8) is a nonlinear, but monotonic transformation of $E[\vec{\varphi}]$, which preserves all of its properties relevant for optimization, i.e., it has the same critical points as $E[\vec{\varphi}]$ and the same relative ordering of the local and global minima. Additionally, it works like a filter by flattening only the portions of the energy surface $E[\vec{\varphi}]$ which lie above $E[\vec{\varphi}^*]$ and leaves it nearly unmodified elsewhere. The term $\sum_{ab}(\varphi_{ab} - \varphi^*_{ab})^{4/3}$ is referred to as the "repeller energy term," which creates a convex surface with a single minimum located at $\vec{\varphi} = \vec{\varphi}^*$. In effect, as seen in Figure 3.2, $C[\vec{\varphi}, \vec{\varphi}^*]$ of (3.8) transforms the current local minimum of $E[\vec{\varphi}]$ into a global maximum such that gradient descent can escape from it to a lower valley.

Thus, when the gradient descent in equation (3.6) is replaced by the Global Descent of (3.7), the Backpropagation algorithm escapes the encountered local minimum of the multidimensional functional $E[\vec{\varphi}]$ of (3.3), due to the following characteristics of the Global Descent system.

The dynamical system (3.7) autonomously switches between the following two phases:

Phase I. This phase, which is effectively a tunneling phase, is characterized by $E[\vec{\varphi}] \geq E[\vec{\varphi}^*]$. Since for this condition the subenergy gradient magnitude is nearly zero in the vicinity of the local minimum $(\vec{\varphi}^*)$, the dynamical system (3.7) behaves approximately as:

$$\dot{\varphi}_{ab} \simeq \eta\, k\, (\varphi_{ab} - \varphi^*_{ab})^{1/3}\ .$$

This system represents a terminal repeller, originally introduced by Zak [Zak89], which has a repelling unstable equilibrium point at $\varphi_{ab} = \varphi^*_{ab}$, i.e., at the local minimum of

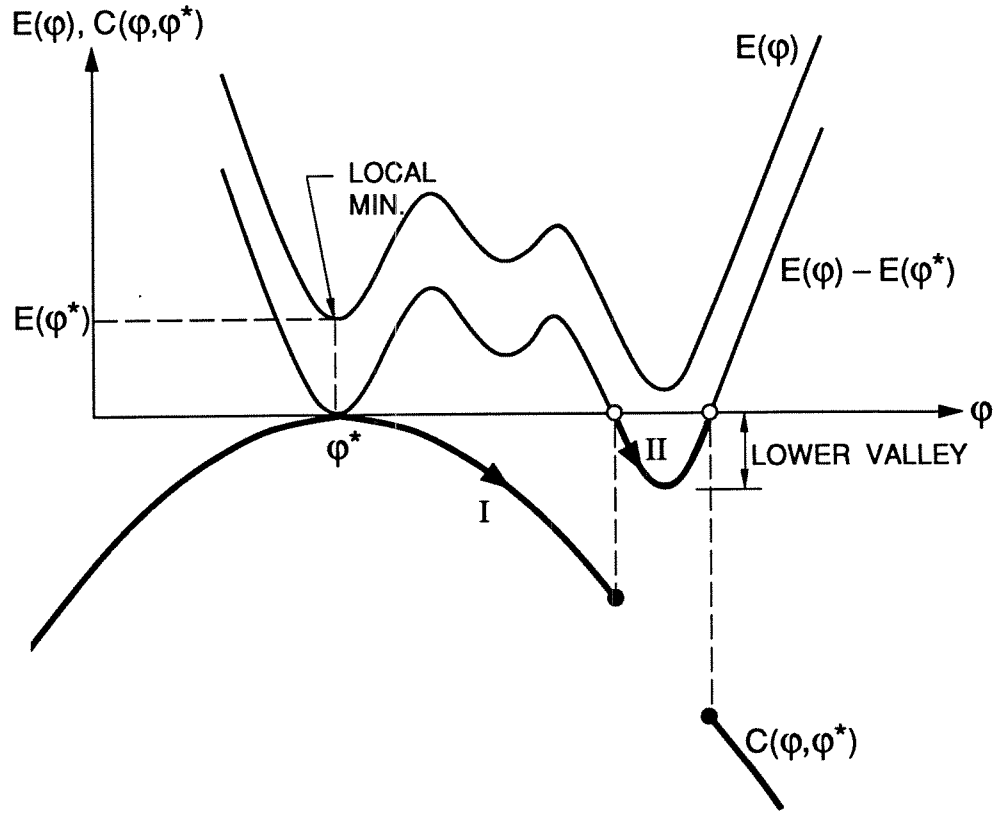

Figure 3.2: A one-dimensional schematical representation of the cost function $C[\varphi, \varphi^*]$ for an arbitrary error energy $E[\varphi]$.

the energy functional $E[\vec{\varphi}]$. Thus, due to this repeller, the dynamical system (3.7), when initialized with a small perturbation from $(\vec{\varphi}^*)$, will be repelled from the local minimum until it reaches a lower basin of attraction, where $E[\vec{\varphi}] < E[\vec{\varphi}^*]$. In effect, this phase tunnels through portions of $E[\vec{\varphi}]$ where $E[\vec{\varphi}] \geq E[\vec{\varphi}^*]$.

Phase II. This phase, which is a minimization phase, is characterized by $E[\vec{\varphi}] < E[\vec{\varphi}^*]$. The gradient multiplier term has approximately unit magnitude, and the repeller term is identically zero. Thus (3.7) behaves as:

$$\dot{\varphi_{ab}} \simeq -\eta \frac{\partial E[\vec{\varphi}]}{\partial \varphi_{ab}}.$$

Clearly this phase implements minimization via gradient descent.

In order to provide a clearer picture of the Backpropagation with Global Descent, we outline its implementation in terms of a step-by-step procedure. First let us assume the two-layered feed-forward network has K units, with M input-to-hidden weights and N hidden-to-output weights, and will be trained for p input-output patterns.

1. Calculating the activation of the units, defining the error energy and finding the energy gradients will be performed using equations (3.1) through (3.5) as in standard Back-propagation. However, training will be performed by using Global Descent dynamics in (3.7), which will be substituted for the former gradient descent law of equation (3.6).

2. To initiate the dynamics, we pick an arbitrary domain in the form of hyper-parallelpiped of dimension $M + N$, and choose $(\vec{\varphi}^*)$ as one corner of the domain. In effect, a repeller is placed at $(\vec{\varphi}^*)$ and the dynamical system in (3.7) is given initial conditions $(\vec{\varphi}^* + \vec{\epsilon}_\varphi)$ where $\vec{\epsilon}_\varphi$ is a small perturbation which drives the system into the domain of interest in weight space.

3. If $E[\vec{\varphi}^* + \vec{\epsilon}_\varphi] < E[\vec{\varphi}^*]$, the system immediately enters a gradient descent phase (**phase II** above), which equilibrates at a local minimum $(\vec{\varphi}^{1*})$.

4. We then set $(\vec{\varphi}^*) = (\vec{\varphi}^{1*})$, and perturb $(\vec{\varphi})$ to $(\vec{\varphi}^{1*} + \vec{\epsilon}_\varphi)$. Note that consistency in the flow direction is necessary.

5. Since $(\vec{\varphi}^{1*})$ is a local minimum, $E[\vec{\varphi}] \geq E[\vec{\varphi}^{1*}]$ holds in a neighborhood of $(\vec{\varphi}^{1*})$. Thus, the system enters the repelling phase (**phase I** above), and the hyperplane repeller located at $(\vec{\varphi}^{1*})$ repels the system until it reaches a lower basin of attraction, where $E[\vec{\varphi}] < E[\vec{\varphi}^{1*}]$. In effect, this phase tunnels through all of the state space region with error energy values that lie above the last found lower local minimum. As the dynamical system enters the next basin, the algorithm automatically switches to gradient descent, leading to minimization of $E[\vec{\varphi}]$ in (3.3). By using the energy gradient flow, the system will equilibrate at the next lower local minimum, $(\vec{\varphi}^{2*})$. We then set $(\vec{\varphi}^*) = (\vec{\varphi}^{2*})$ and repeat the process.

6. If $E[\vec{\varphi}^* + \vec{\epsilon}_\varphi\,] \geq E[\vec{\varphi}^*\,]$ when the training is initiated, (3.7) is initially in a tunneling phase. The tunneling will proceed to a lower basin, at which point it enters the minimization phase and follows the behavior discussed above.

7. The successive minimization and tunneling computational processes continue until a suitable stopping criterion is satisfied. If an exact solution for the weights exist such that for the specific problem input patterns can be accurately mapped to the target output patterns, then the global minimum at $E[\vec{\varphi}] = 0$ will set the stopping criteria. However if an exact solution does not exist, the global minimum associated with $E[\vec{\varphi}] > 0$ can be located by ocular inspection (it is detected as the lowest local minimum) of the energy $E[\vec{\varphi}]$ vs. the number of training epochs curve, which will indicate the optimal solution to the problem. Additionally the global minimum satisfies the criteria of a local minimum; that is, $\sum_{\varphi_{ab}} (\partial E[\vec{\varphi}^*]/\partial\varphi)^2 \simeq 0$ (i.e., energy gradients for all weights being close to zero).

It is worth noting that training the network in the batch mode (i.e., weights are adjusted after presenting all input-output patterns) is required, since the system in (3.7) necessitates a unique energy surface $E[\vec{\varphi}]$ of (3.3) for all patterns.

Evidently, the structure of the dynamical system in (3.7) is highly parallel, allowing implementation in a form whose computational complexity is only weakly dependent on problem dimensionality (i.e., $M + N$ here).

3.3 Examples and Applications

This section presents results of benchmark tests carried out for comparison of standard Backpropagation and Backpropagation associated with Global Descent. Our first two examples are problems that are often used for benchmarking a network [HKP91]: the XOR and the parity problem. We also consider a pattern recognition example of 60 input-output patterns as an application. In all of them we demonstrate that for some initial weights, learning with the standard Backpropagation gets caught in a local minimum and either

Table 3.1: Input, target, output results and local minimum weight states for XOR function

pattern #	ξ_1	ξ_2	Target	Local	Global
1	0.0	0.0	0.0	0.056	0.030
2	0.0	1.0	1.0	0.961	0.963
3	1.0	0.0	1.0	0.493	0.964
4	1.0	1.0	0.0	0.498	0.031

Local Minimum Weights		
$w_{10} = 1.3643$	$w_{20} = 5.5906$	$v_{10} = 9.7378$
$w_{11} = -18.58$	$w_{21} = 19.143$	$v_{11} = -10.88$
$w_{12} = -5.4145$	$w_{22} = -7.6285$	$v_{12} = -9.7378$

gives an inferior solution or even may be unable to solve the problem. Backpropagation with Global Descent on the other hand escapes the encountered local minima and converges to the global minimum of the error energy function. For simplicity we will refer to the former as gradient descent and latter as Global Descent.

3.3.1 The XOR Problem

We begin with the exclusive-or problem since it is a classical problem requiring hidden units and since many other difficult problems involve an XOR as a subproblem. The truth table with the input and target patterns of the XOR function is shown in the first four columns of Table 3.1. One of the common problems in doing the XOR problem with a standard back-error paradigm is the presence of the local minima [Day90, Blu89].

We employed the smallest number of units in the network that can accomplish the XOR function: two units in the hidden layer, one in the output layer. The network required two external inputs, and bias units as demonstrated in Figure 3.1. The network with nine connection weights presented a 9-dimensional optimization problem. 50% of all simulations with random initial weights got stuck in a local minimum when using gradient descent. Figure 3.3 shows a comparison simulation of gradient descent and Global Descent for a specific random initial weight set. As the dashed line indicates, gradient descent got caught

in a local minimum causing 12.74% error (i.e., 0.255/2.0) and thus evaluated only the first two entries in the training set correctly and failed for others as indicated in the fifth column of Table 3.1. Application of more training iterations in this case failed to get better convergence. Global Descent, on the other hand, as the solid line of Figure 3.3 shows, escaped the local minimum by tunneling through functionally higher values of the error function (seen as hill) and located the global minimum with only 0.11% error and solved the XOR problem as demonstrated in the sixth column of Table 3.1.

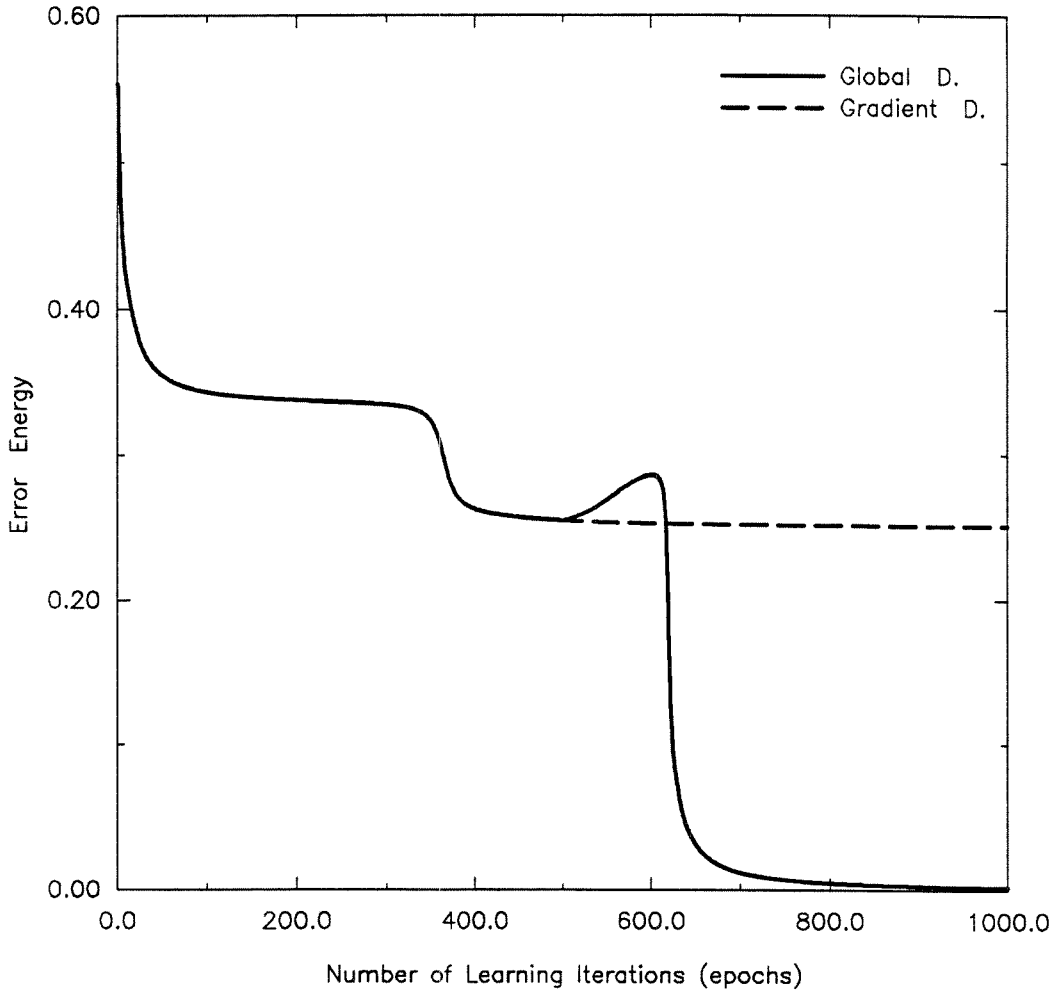

Figure 3.3: Comparison of Global vs Gradient Descent for XOR function.

It is also worth noting that gradient descent fails to escape the critical point even after four million training iterations. Furthermore, to disprove the popular speculation that the XOR and similar functions do not have local minima but rather possess almost flat regions with extremely small slopes [MHB89], we have calculated the eigenvalues of the Hessian matrix for the local weights shown in Table 3.1, and found that they all have positive values, which proves the existence of a true local minimum.

3.3.2 The Parity Problem

The parity problem is essentially a generalization of the XOR problem to K inputs, and has been extensively discussed by Minsky and Papert [MP69]. The single output unit is required to be on if an odd number of inputs are on, and off otherwise. It is often used for evaluating network performances and has been classified as a challenging problem, since the output changes whenever any single input unit changes, i.e., the most similar input patterns correspond to different target patterns. This fact has been observed to cause local minima for certain initial weights.

The network topology has four inputs, four units in the hidden layer and and a single output unit. Thus training the network gives rise to a 25-dimensional optimization problem, (i.e., $E[\vec{\varphi}]$ in (3.3) contains 25 states as the connection weights). The training set has 16 input-target patterns. Figure 3.4 shows a performance comparison between the aforementioned methods for a specific random initial weight set. While gradient descent produced an unacceptable local solution with a 17.59% error, Global Descent after escaping through two consecutive local minima converged to the global solution with 0.22% error.

3.3.3 A Pattern Recognition Example

We also considered a simple pattern recognition example of 60 patterns [Ati91]. Figure 3.5 shows a set of two-dimensional training patterns from three classes. This requires the design of a neural network recognizer with three output neurons, each of which shall be on if the coordinates of a sample of the corresponding class is presented, i.e., a "winner take all"

network is necessary. Several simulations suggested that at least two units in the hidden layer were required to solve the problem. Figure 3.6 presents our results. Gradient descent trained the network to learn the patterns with 13.77% (i.e., 12.4/90) error due to a local minimum in the 15-dimensional error energy. Performing the same experiment with Global Descent solved the problem globally with a 0.32% total error for all 60-patterns.

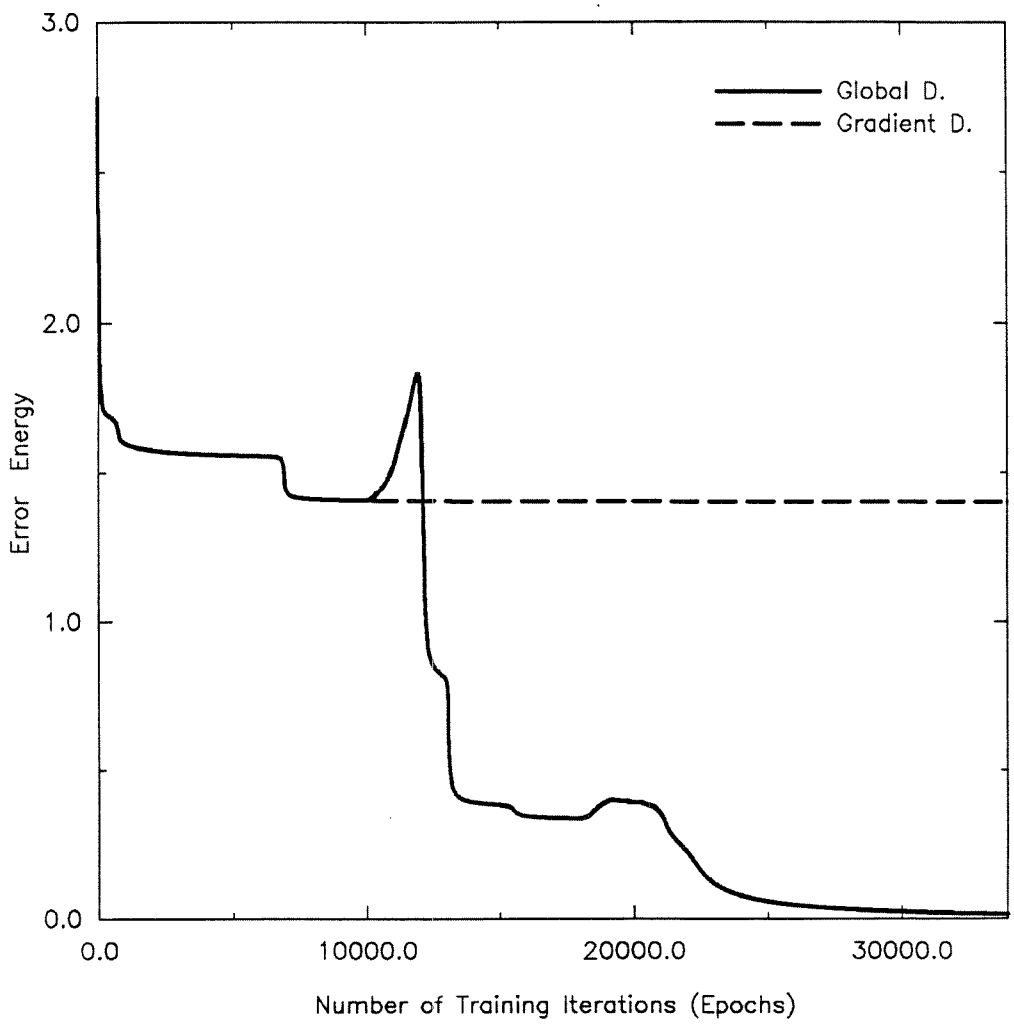

Figure 3.4: Comparison of Global vs Gradient Descent for Parity function.

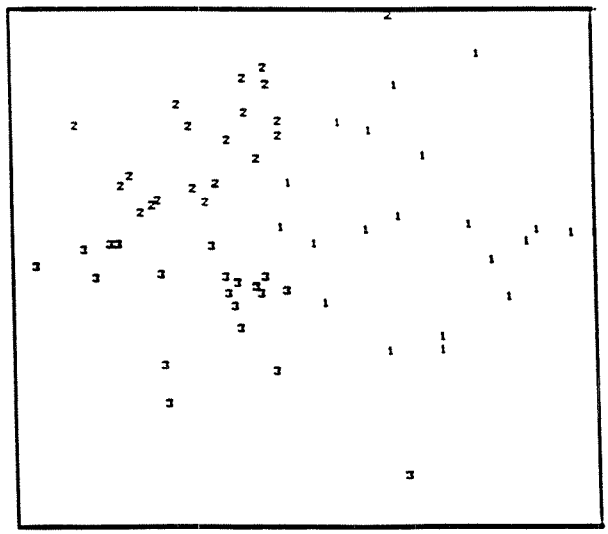

Figure 3.5: Training patterns for Pattern Recognition

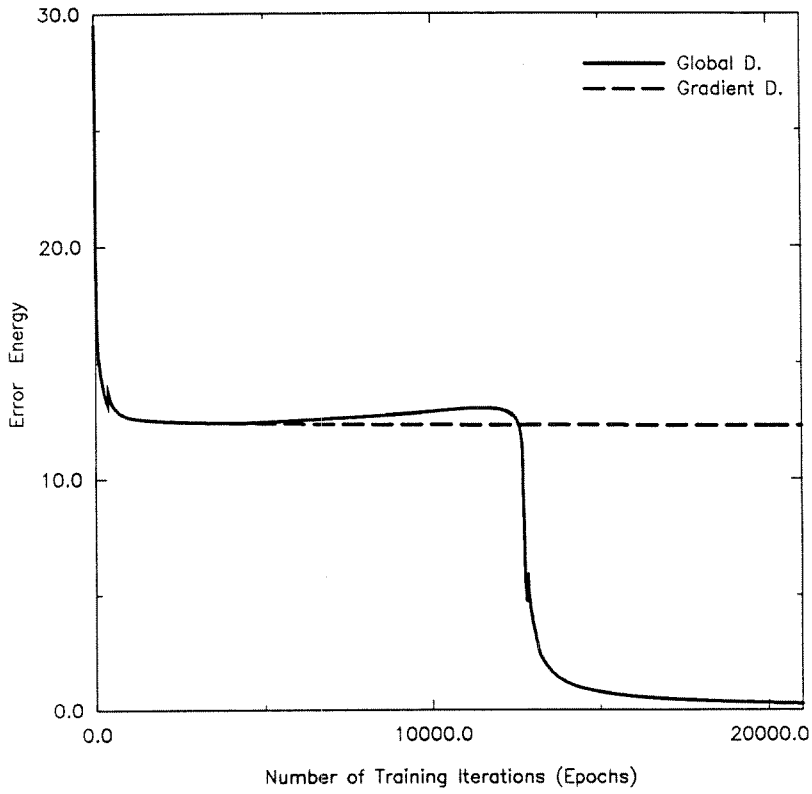

Figure 3.6: Comparison of Global vs Gradient Descent for Pattern Recognition.

3.3.4 Discussion

Two other benchmark tests are carried out with less success than those mentioned above. These are the chaotic time series and the binary addition problem. In the first test, no local minima was found for any initial condition, so that Global Descent performed identical to Gradient Descent. In the second test, for some initial conditions, Global Descent failed to converge to the global minimum. However, solutions that are very close to the optimal solution were found.

It is important to note that in all the simulations above, we mainly concentrated on the local minima problem. The issue of rate of convergence was not the objective of this thesis, therefore we used the simple Euler integration scheme which resulted in a relatively large number of iterations (i.e., epochs). Better integration schemes can speed up convergence.

3.4 Summary

In this chapter, we have introduced Global Descent in the context of artificial neural networks to improve learning. The methodology, which is based on the TRUST algorithm of Chapter 2, is proposed as a candidate for replacing the gradient descent formalism in the Backpropagation algorithm, in order to eliminate the local minima problem during training.

Benchmark tests demonstrate that Global Descent associated with Backpropagation escapes encountered local minima, and in most cases converges to the globally optimal solution of a specific problem.

Chapter 4 discusses an Analog VLSI implementation of the TRUST algorithm and also presents the results of the measured and simulated experiments performed with the circuits, we have designed.

Chapter 4

Analog VLSI Circuits for Global Optimization

This chapter reviews the design of a novel circuit that implements the TRUST algorithm (or Global Descent) for an arbitrary one-dimensional objective function. The global circuit consists of two main modules, which implement the concepts of terminal repeller and gradient descent, and an auxiliary module that unifies these two concepts. First, the main modules are considered in detail, and then the entire global optimization circuit is analyzed.

4.1 Terminal Repeller Circuit

As discussed in Subsection 2.3.2, terminal dynamical systems are based on the non-Lipschitzian dynamics of odd power-law transfer characteristics and have unusual behavior at the terminal point. The properties of terminal repellers can be illustrated by the example

$$\dot{x} = x^{1/3}. \tag{4.1}$$

As Figure 4.1 suggests, terminal systems in their phase space can be observed to have transfer functions (i.e., $\dot{x}$ vs x), which are odd-symmetric and have a very high slope (ideally

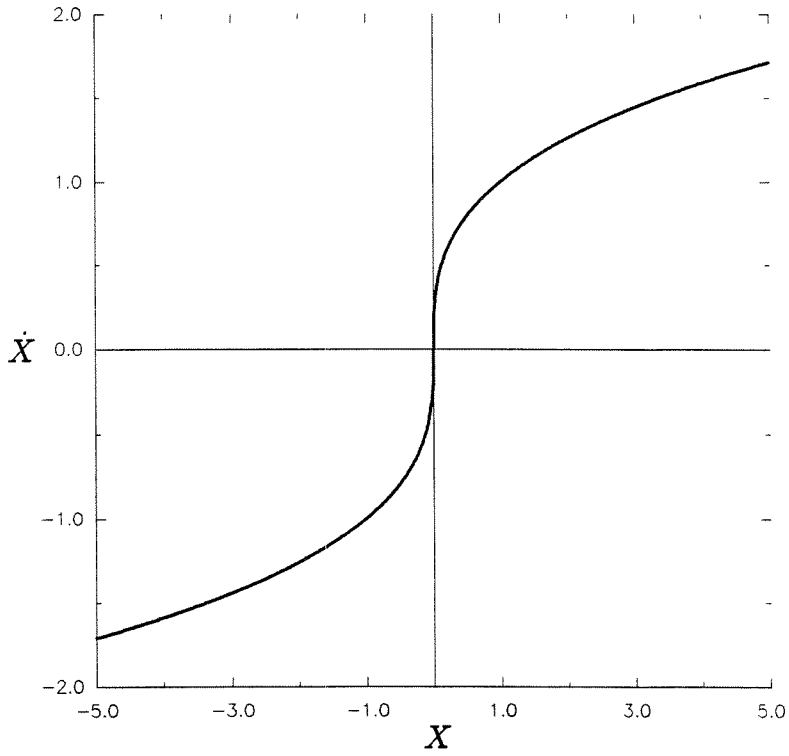

Figure 4.1: Terminal repeller phase space diagram

infinite) at the terminal point. However, designing an exact cubic root circuit would be impossible due to the analog circuit components, which precludes attaining integer numbers. We therefore implement the following system to obtain the behavior of (4.1):

$$\dot{x} = \begin{cases} x^{1/m}, & x \geq 0 \\ -(-x)^{1/m}, & x < 0 \end{cases} \tag{4.2}$$

where, $m > 0$ is a positive real number. Equation (4.2) preserves the exact terminal characteristics that (4.1) has.

The term $x^{1/m}$ for $x > 0$ can be obtained by employing Mead's square root circuit [Mea89], which is shown in Figure 4.2. In this circuit, the input current can be written by using the transistor characteristics,

$$I_{in} = I_0 \, e^{\kappa V_a - V_1} \tag{4.3}$$

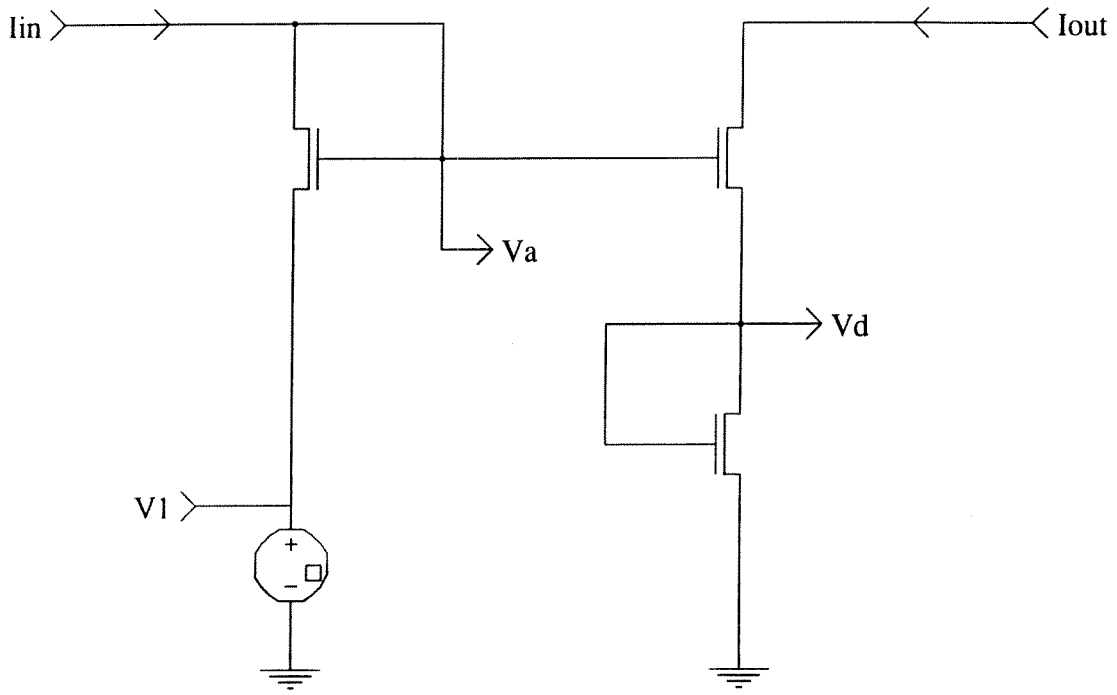

Figure 4.2: Square root circuit

and, the output current can be found similarly,

$$I_{out} = I_0\, e^{\kappa V_a - V_d} \;=\; I_0\, e^{\kappa V_d}. \tag{4.4}$$

Thus, V_a can be solved from (4.4) and substituted into (4.3) to yield,

$$I_{in} = I_0\, e^{(\kappa+1)V_d - V_1} \tag{4.5}$$

where, V_d is given by (4.4), such that equation (4.5) becomes,

$$I_{in} = I_0\, e^{-V_1} \left(\frac{I_{out}}{I_0}\right)^{(\kappa+1)/\kappa}. \tag{4.6}$$

Finally, solving (4.6) for I_{out} yields,

$$I_{out} = c_1\, e^{(\kappa/\kappa+1)V_1}\, (I_{in})^{\kappa/\kappa+1} \tag{4.7}$$

where $c_1 = I_0^{1/\kappa+1}$, I_0 is the leakage current, and I_{in} and I_{out} are positive valued input and output currents respectively. Note that for $\kappa = 0.7$, the exponent $\kappa/(\kappa+1)$ is $1/2.43$.

The transfer function of equation (4.7) is shown in Figure 4.3 for different values of V_1.

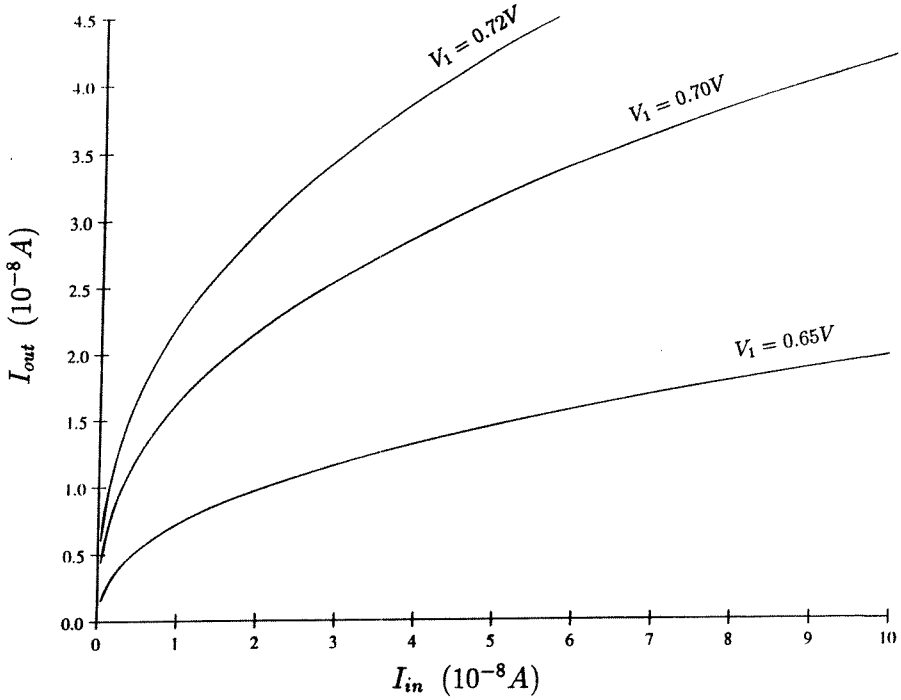

Figure 4.3: Transfer characteristics for square root circuit of Figure 4.2 for various V_1.

As Figure 4.3 suggests, V_1 is a gain factor in the square root circuit.

However, the square root circuit of Figure 4.2, having the dynamics of (4.7), operates only in the first quadrant, i.e., for positive currents. To implement the term $-(-x)^{1/m}$ for $x < 0$ of (4.2), i.e., operation in the third quadrant, we implemented a mirror image of the square root circuit by replacing the n-channel transistors by p-channel transistors in a manner given in Figure 4.4.

A similar analysis yields the following transfer function for this circuit,

$$-I_{out} = c_2\, e^{(\kappa/\kappa+1)V_2}\,(-I_{in})^{\kappa/\kappa+1} \tag{4.8}$$

where, c_2, similar to c_1, is a constant that depends on I_0; I_{in} and I_{out} are negative valued

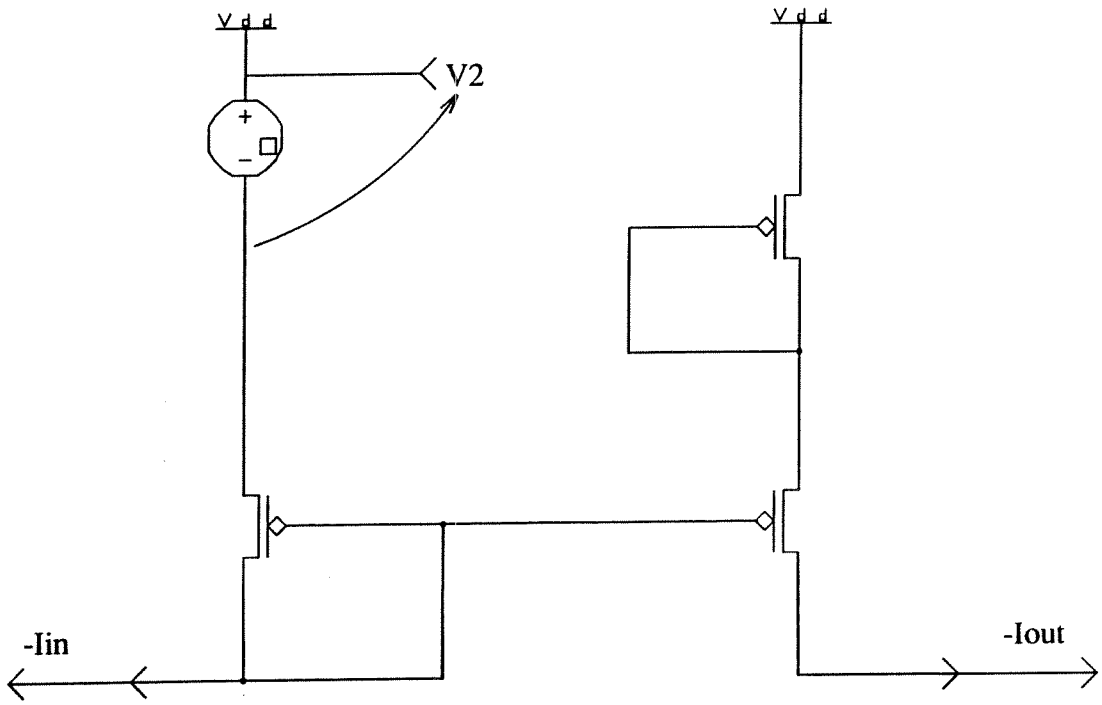

Figure 4.4: Square root circuit for negative currents

currents, and V_2 is a constant gain voltage.

Finally, the repeller transfer function characteristics of (4.2) can be achieved by employing both circuits of Figure 4.2 and Figure 4.4 as shown in Figure 4.5.

The symmetry in this circuit enables operation in the first and third quadrants. That is, when $I_{in} > 0$, transistor Q_1 is on and Q_7 is off, resulting in $I_{in} = I_1$. When $I_{in} < 0$, th condition reverses to yield $I_{in} = -I_2$. Note that I_1 and I_2 are the input currents of the squar root circuits, which operate in the first and third quadrant respectively. Furthermore, th current mirrors, at the end of the circuit, superimpose both quadrants with $I_{out} = I_5 - I_6$ where I_5 and I_6 are the output currents of the square root circuits operating in the first and third quadrant respectively. Thus, the circuit of Figure 4.5 implements the following transfer function,

$$I_{out} = \begin{cases} c_1\, e^{(1/2.43)V_1}\, (I_{in})^{1/2.43}, & I_{in} > 0 \\ -c_2\, e^{(1/2.43)V_2}\, (-I_{in})^{1/2.43}, & I_{in} < 0 \end{cases} \tag{4.9}$$

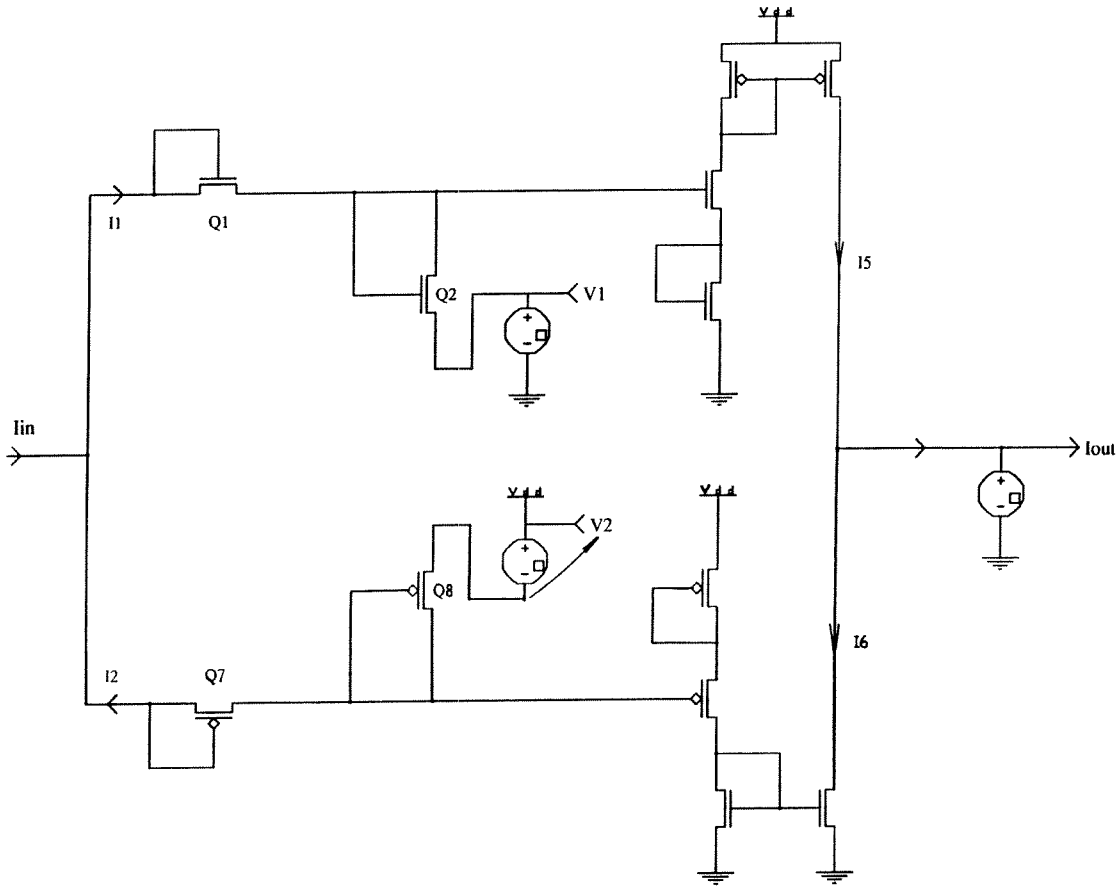

Figure 4.5: Terminal repeller transfer function circuit

which has the desired form of equation (4.2), with I_{out} and I_{in} corresponding to $\dot{x}$ and x respectively.

We have fabricated the circuit of Figure 4.5 and tested its performance by sweeping the input current source with $1nA$ intervals and measuring the corresponding output current via a current meter. The result of the experiment is shown in Figure 4.6, which reveals the measured DC transfer function.

Note that the circuit captures the important characteristics of the terminal systems, that is, having odd-symmetry around the origin and a very high vertical slope in the neighborhood of the origin, as shown in Figure 4.7.

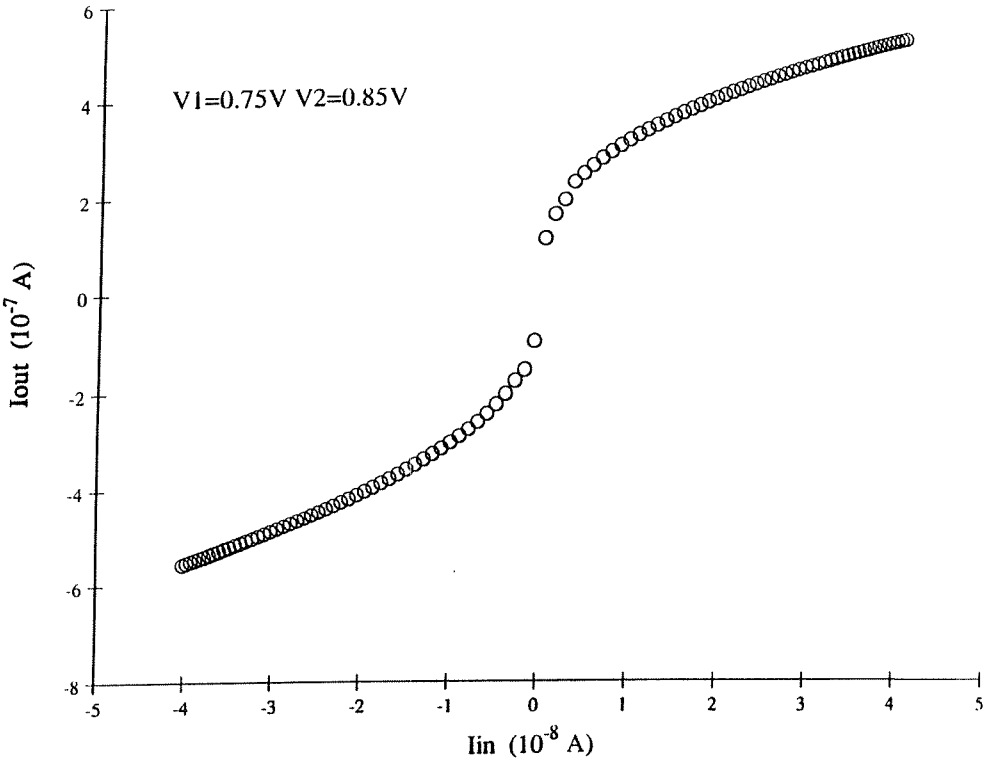

Figure 4.6: Terminal repeller transfer function data as measured from the chip.

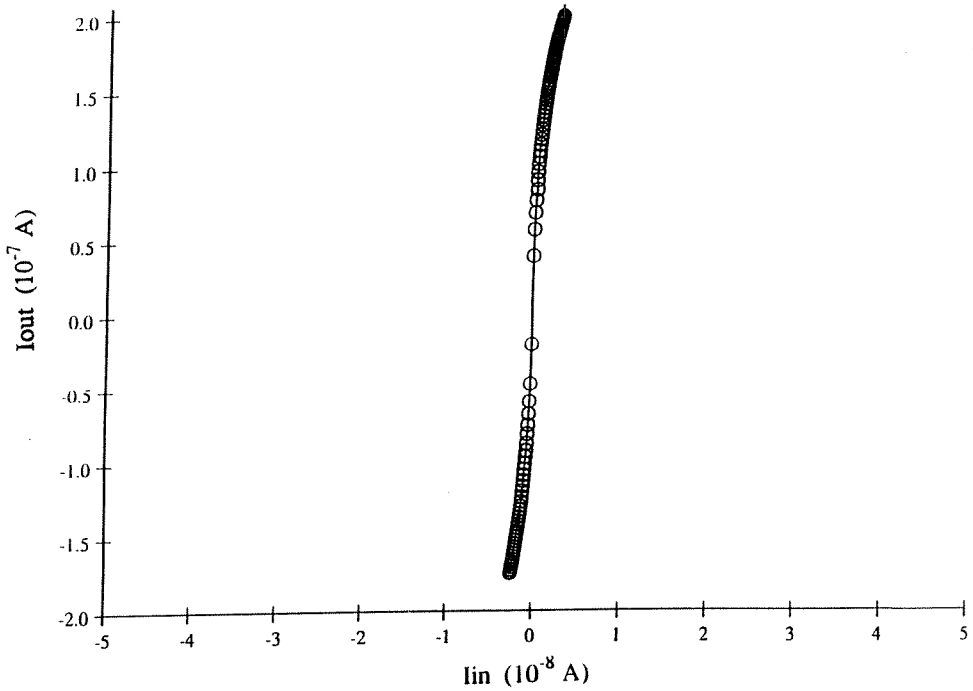

Figure 4.7: Measured transfer function characteristics of the circuit near zero current.

It is interesting to note that as a by-product we achieved a bidirectional high gain current-mode amplifier.

We have also control over the terminal characteristics, i.e., on the vertical slope of the transfer function by adjusting the knobs V_1 and V_2, which serve as the gain factors. Additionally, setting V_1 and V_2 to a subthreshold level brings the output current I_{out} to a desired level (i.e., at the order of $10^{-9}A$). It should be noted that when V_1 and V_2 are zero, the output current is at the order of femtoamps, which would be negligibly small. However, in order to operate the circuit transistors in the saturated region at all times, the maximum limit for V_1 and V_2 (about $1V$ each) have to be taken into account. Another significant issue is that, although the diodes Q_1 and Q_7 operate as digital switches, they have the additional role of supplying about $1V$ drop across their terminals. Because of this, when the input current is zero, the inner loop of the circuit (i.e., $V_2 \Rightarrow Q_8 \Rightarrow Q_7 \Rightarrow Q_1 \Rightarrow Q_2 \Rightarrow V_1$), which causes a voltage drop of about $5V$ against the upper rail ($V_{dd} = 5V$), will not yield an undesired non-zero output current.

By adding additional components, the terminal repeller transfer function circuit of Figure 4.5 can be extended to achieve the terminal repeller dynamics of equation (4.2). Figure 4.8 shows the terminal repeller dynamics circuit. Note that the capacitor C integrates I_{out} of the repeller transfer function circuit,

$$I_{out} = C\ dX/dt. \tag{4.10}$$

From previous analysis we know that I_{out} is related to I_{in} as in equation (4.9), which for positive I_{in} has the form

$$I_{out} = c_1\ e^{(1/2.43)\,V_1}\ (I_{in})^{1/2.43}. \tag{4.11}$$

For simplicity and clarity we will use (4.11) instead of (4.9), since the corresponding case for negative I_{in} merely requires sign changes in (4.11). Equating equations (4.11) and (4.10) yields

$$\dot{X} = (c_1/C)\ e^{(1/2.43)\,V_1}\ (I_{in})^{1/2.43}. \tag{4.12}$$

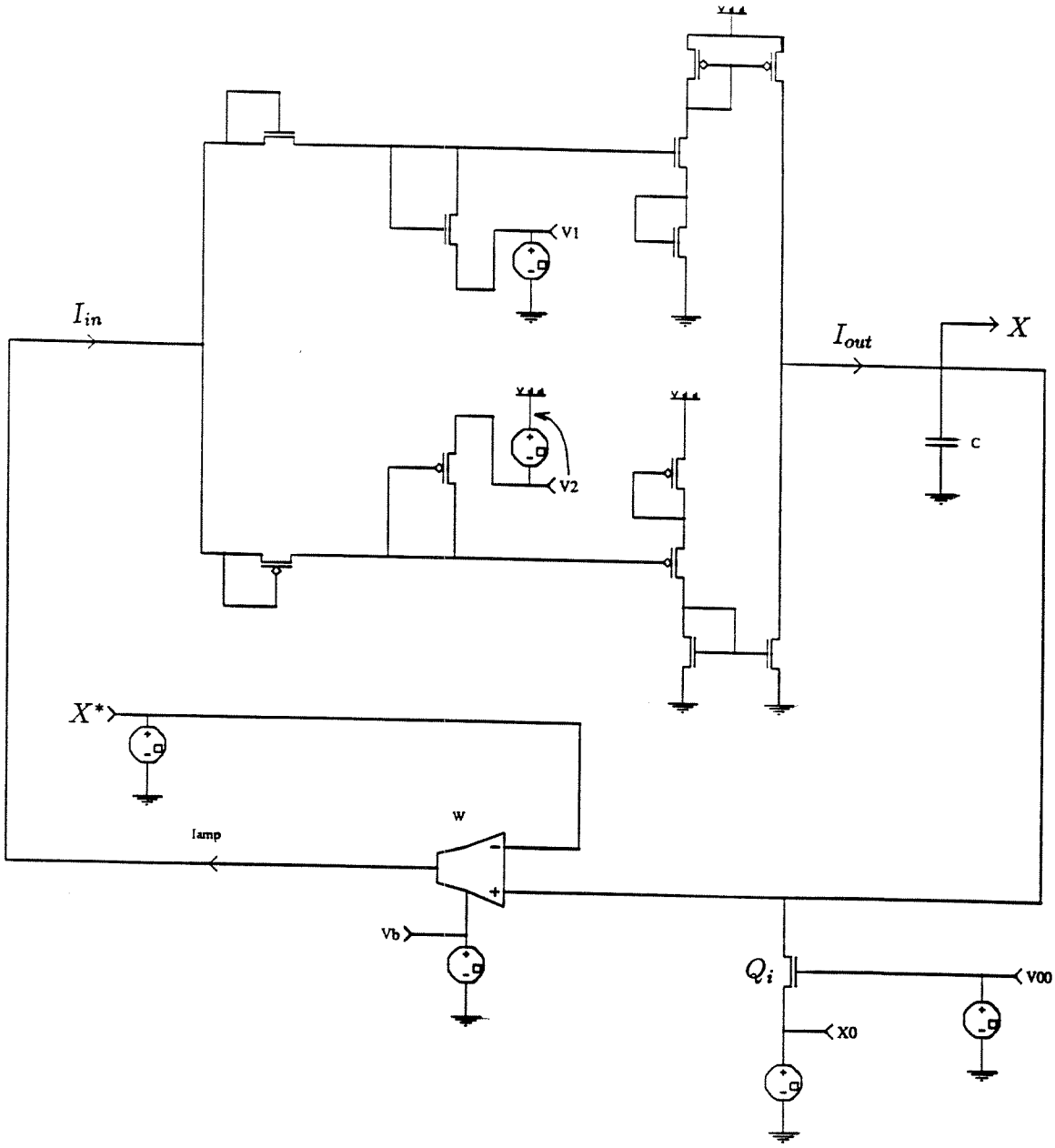

Figure 4.8: Terminal repeller dynamics circuit

The voltage X at the capacitor terminals could not be fed back directly to the input of the terminal repeller transfer function circuit because it requires a current input. Hence, we employ a wide-range transconductance amplifier W to linearly convert the voltage X to current I_{in}, assuming that W is used in its linear range. Thus the amplifier characteristics for small signals (i.e., $\Delta X = X - X^* \tilde{<} 100mV$) can be approximated as

$$I_{amp} = G\,(X - X^*) \tag{4.13}$$

where, $G = \kappa\, I_0\, e^{\kappa V_b}/2\, V_T$ is the transconductance, V_b is the bias voltage, $V_T = 25mV$ is the constant thermal voltage, and X^* is a constant voltage with properties explained below. The amplifier output is fed back to terminal repeller transfer function circuit and therefore simply yields,

$$I_{in} = I_{amp}. \tag{4.14}$$

Finally, manipulating (4.12) with (4.13) and (4.14) results in

$$\dot{X} = K\,(X - X^*)^{1/2.43} \tag{4.15}$$

where, $K = (c_1/C)\, e^{(1/2.43)\,V_1}\, (G)^{1/2.43}$.

Equation (4.15) represents a terminal repeller dynamical system, which has an unstable equilibrium point at $X = X^*$. The initial condition for the dynamical system is supplied by the variable voltage source X_0, which sets X to X_0 when V_{00} is set to 5V such that the transistor Q_i is on and short-circuited. The dynamics of (4.15) is initiated by switching V_{00} from 5V to 0V, which turns off Q_i and thus disconnects the rest of the circuit from X_0. This enables the circuit to start with the initial condition $X = X_0$.

The simulated behavior of this circuit, which implements (4.15), is shown in Figure 4.9 for the initial condition $X_0 = 2.55V$. In this simulation, X^* is set to $2.5V$, placing a repeller there. When the dynamical system is initiated with a small perturbation from the repelling point, it will be repelled from the unstable point to the rail (V_{dd} in this case) in less than 0.1 micro-seconds. Note that K represents the power of the repeller and can be increased

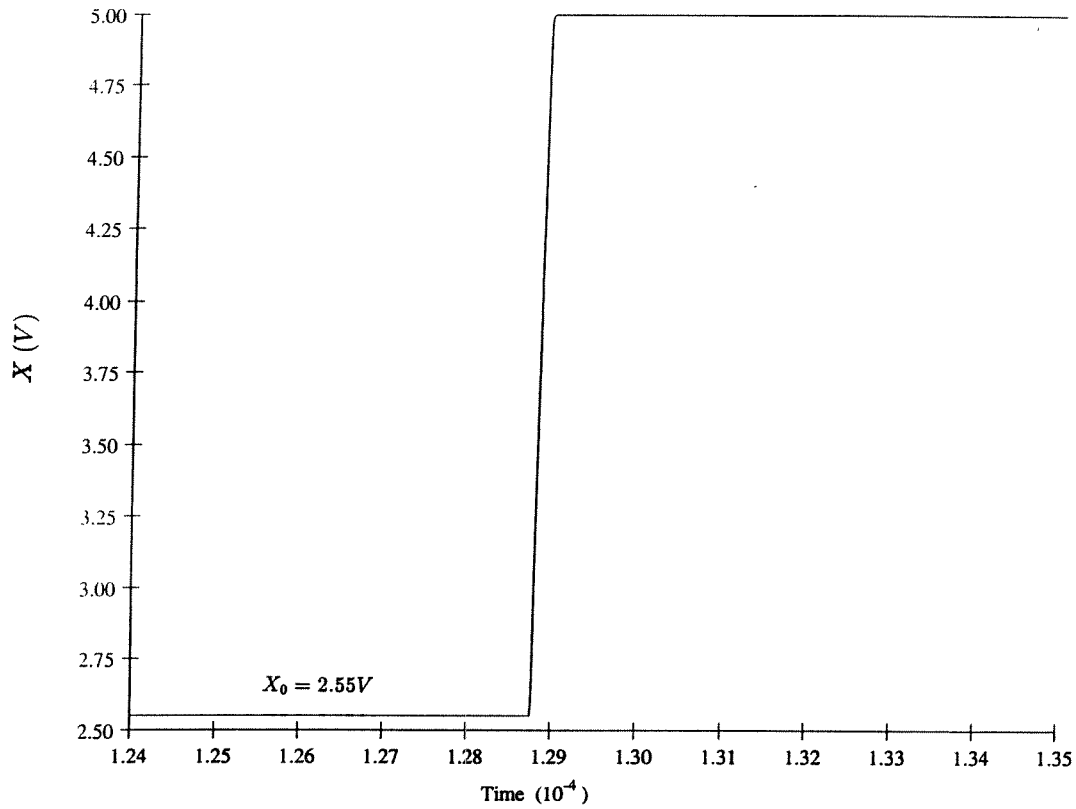

Figure 4.9: Simulation of the repeller circuit response for the initial condition $X_0 = 2.55V$.

by increasing V_1, V_2 or V_b.

The terminal repeller circuit becomes a terminal attractor circuit by switching the inputs of the wide-range amplifier W, so that the new circuit implements

$$I_{amp} = G\,(X^* - X) = -G\,(X - X^*) \tag{4.16}$$

instead of equation (4.13). Therefore combining (4.16) with (4.14) and (4.12) yields,

$$\dot{X} = -K\,(X - X^*)^{1/2.43} \tag{4.17}$$

where K is as given in (4.15).

Equation (4.17) represents a terminal attractor dynamical system, which has a stable equilibrium point at $X = X^*$. We have also designed and fabricated a terminal attractor circuit as described above and as seen in Figure 4.10.

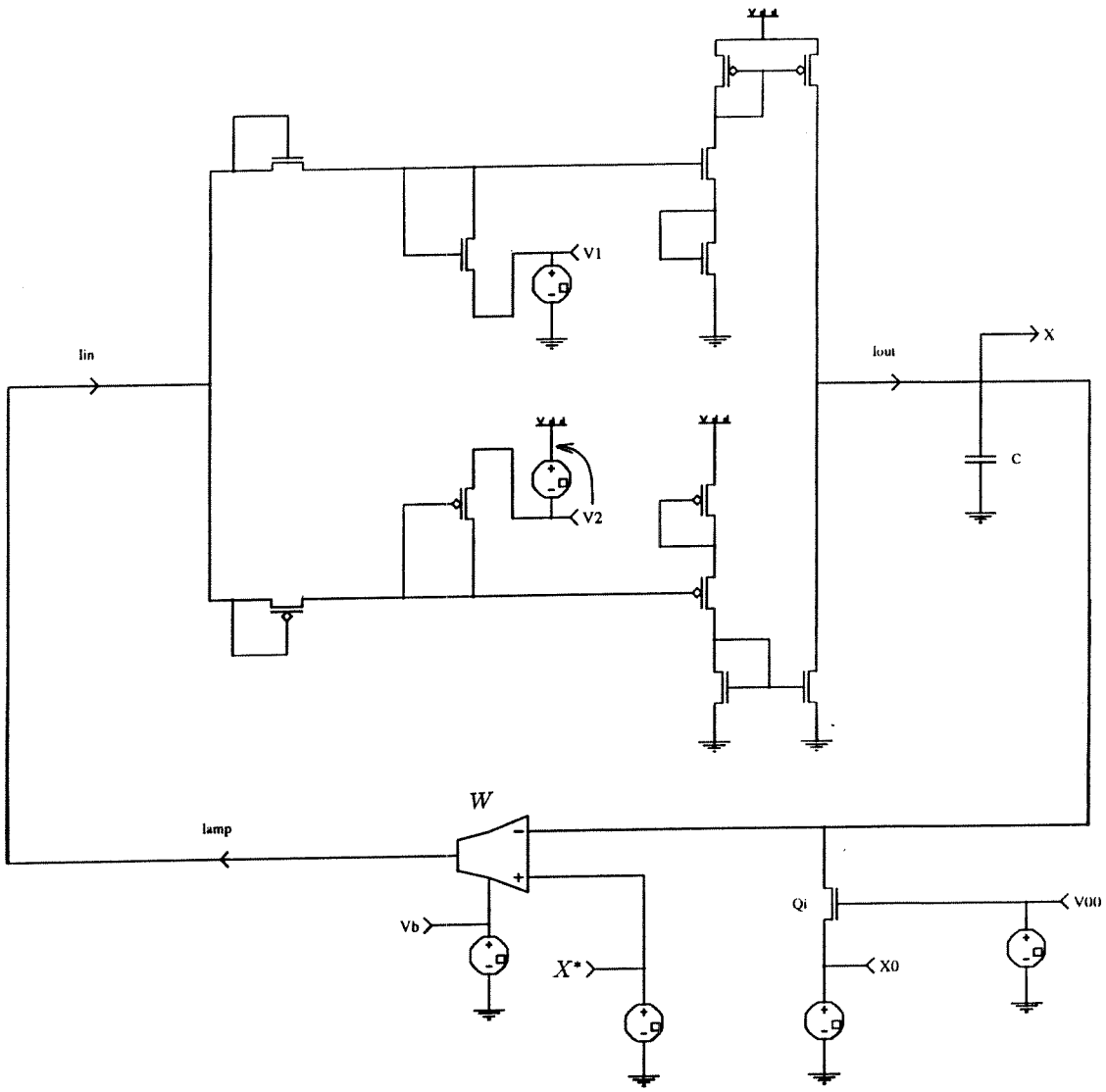

Figure 4.10: Terminal attractor dynamics circuit

The terminal attractor circuit was tested by observing the voltage X on an oscilloscope. The time solution of X is plotted in Figure 4.11 for the initial condition $X_0 = 3.6V$. X^* is set to $2.5V$, placing an attractor there. As Figure 4.11 suggests, the voltage X of the terminal attractor circuit, starting from an initial point, tends to and intersects with equilibrium point at $X = X^*$. The behavior is neither exponential nor underdamped in accordance

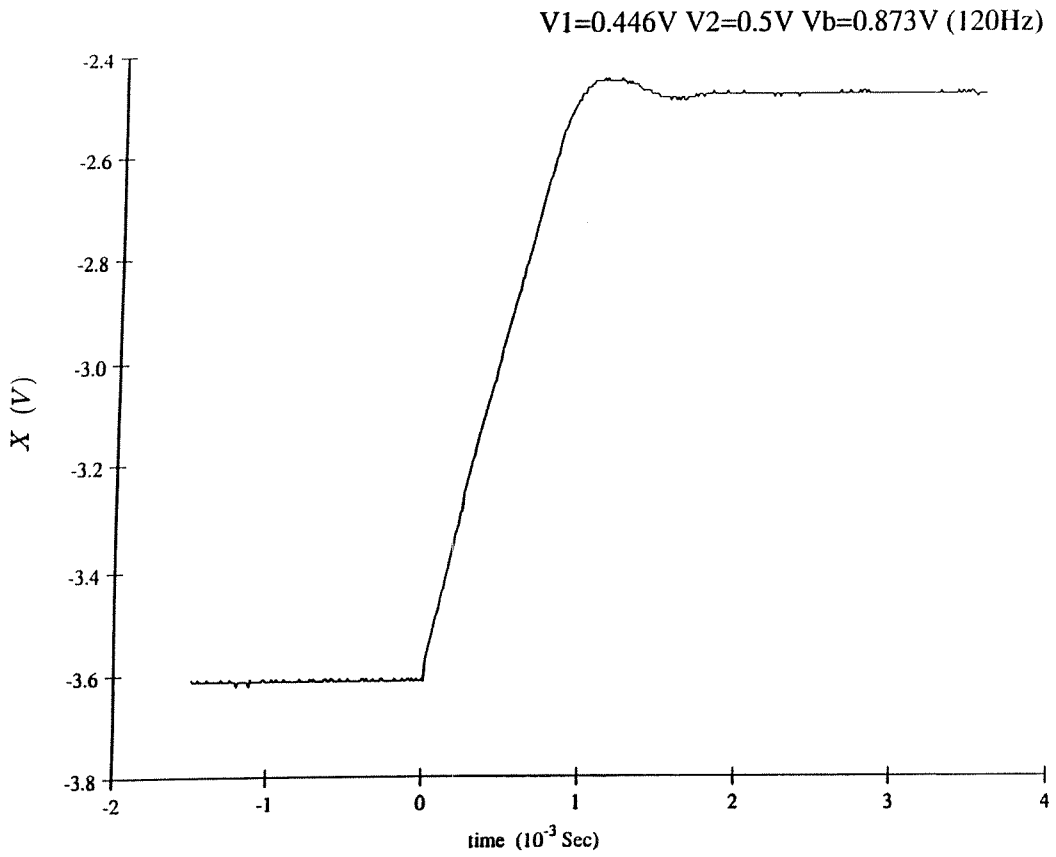

Figure 4.11: Experimental response of the terminal attractor circuit for the initial condition $X_0 = 3.6V$ and attractor at $X^* = 2.5V$.

with the desired terminal characteristics, i.e., the dynamics show first-order differential equation characteristics, but also intersects the solution in a finite time. However, there exists a small bump when the solution intersects the equilibrium point, which is due to internal parasitic capacitances inside the transistors, which tend to create a second order behavior. Such behavior is nevertheless negligible, since it dies out in about 400 micro-seconds. Furthermore, such parasitic capacitances can be made negligibly small compared to C, by choosing a larger value for capacitance C. In above experiments C was set to $1pF$.

Thus, we conclude that the circuit of Figure 4.9 is a first-order terminal attractor circuit, whose behavior is close to that of the ideal terminal attractor.

4.2 Gradient Descent Circuit

Gradient descent can be used to locally minimize a given objective function. Thus, it is useful for a wide range of optimization applications and its role in learning with artificial neural networks was discussed in Section 3.1.

In this thesis, gradient descent is a major component of the global optimization circuit. We have developed, fabricated and tested a circuit for this purpose, which will be considered in this section.

Other investigators have also developed gradient descent circuits. Figure 4.12 shows the conceptual scheme for implementing a gradient ascent circuit, that was suggested by Umminger and DeWeerth [UD89].

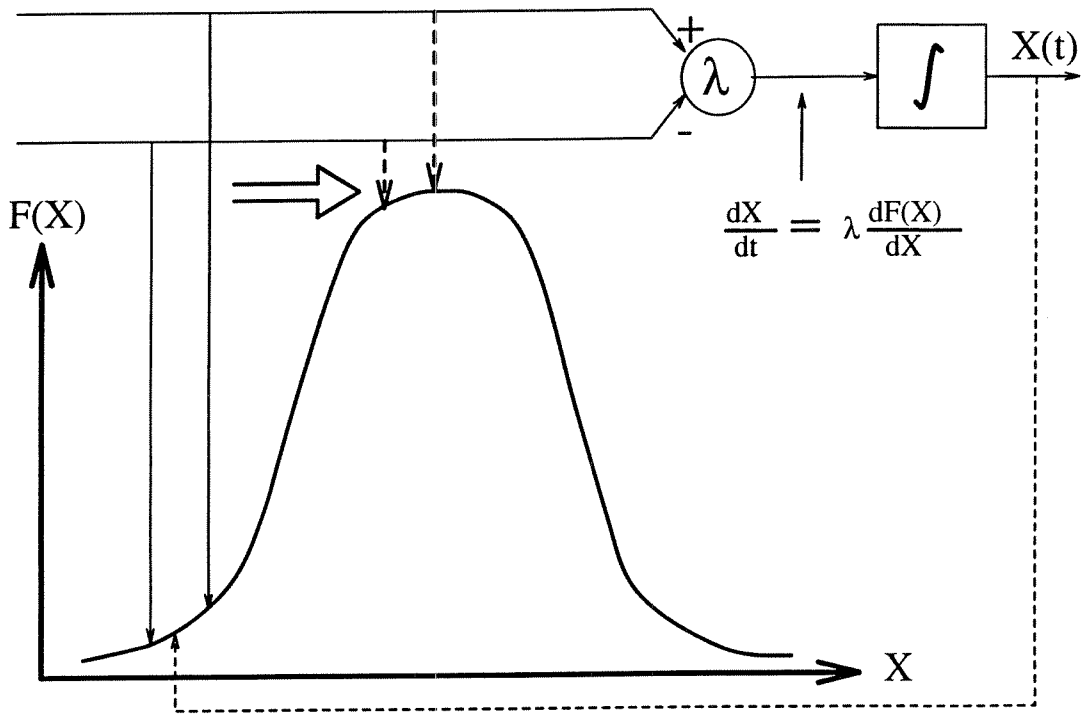

Figure 4.12: Schematical representation for implementing gradient ascent by Umminger et al.

Although our circuit is different than this prior effort, it uses the same conceptual approach, which is based on taking two sample points from an objective function (or energy function), comparing the corresponding energy state, multiplying the difference by a fixed constant λ, integrating the end-product, and feeding back this result to the input in order to advance the two sampling points one step further. In this way, the gradient descent rule $\dot{X} = -\lambda\ dF(X)/dX$ is expected to converge to the nearest local minimum of $F(X)$ (see gradient ascent in Figure 4.12).

The drawback of the aforementioned "gradient descent scheme" is that it samples the objective function discretely rather than smoothly. More explicitly this means, the gradient descent

$$\dot{X} = -\frac{dF(X)}{dX} = -\lim_{\Delta X \to 0} \frac{F(X + \Delta X) - F(X)}{\Delta X} \tag{4.18}$$

is approximated by the following formulation:

$$\dot{X} \simeq -\frac{F(X + \Delta X) - F(X)}{\Delta X} \tag{4.19}$$

where, ΔX is a small constant. The equilibrium point of this system is close to, but not exactly a local minimum.

Figure 4.13 shows our Gradient Descent circuit, which mainly consist of an objective function block, an integrator, and an initial condition generator.

A wide class of objective functions with adjustable location and steepness of the minima are implemented using a cascade of voltage correlator bump circuits. The fabricated circuit (Figure 4.13) includes three blocks of such bump circuits, which are cascaded at the bottom layer of Figure 4.13. The characteristic equation of each bump circuit is,

$$I_{out} = \frac{I_b}{4}\ \mathrm{sech}^2\ \frac{\kappa(X - X_{ref})}{2}. \tag{4.20}$$

The center of each bump is located at $X = X_{ref}$, as seen in Figure 4.14, while the steepness of the bump depends on the value of the bias voltage V_b (i.e., $I_b \simeq I_0\, e^{\kappa V_b}$).

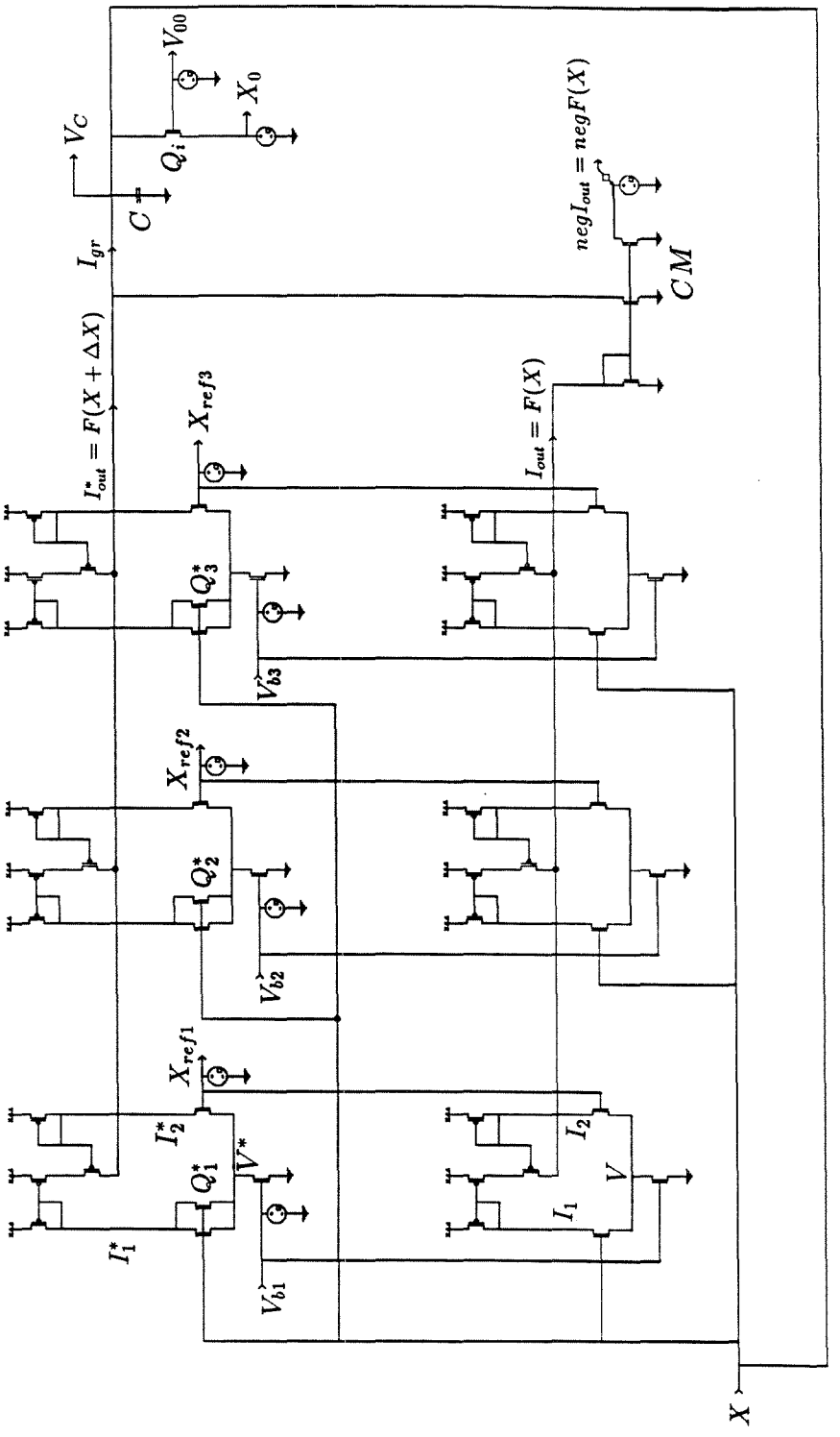

Figure 4.13: Gradient descent circuit

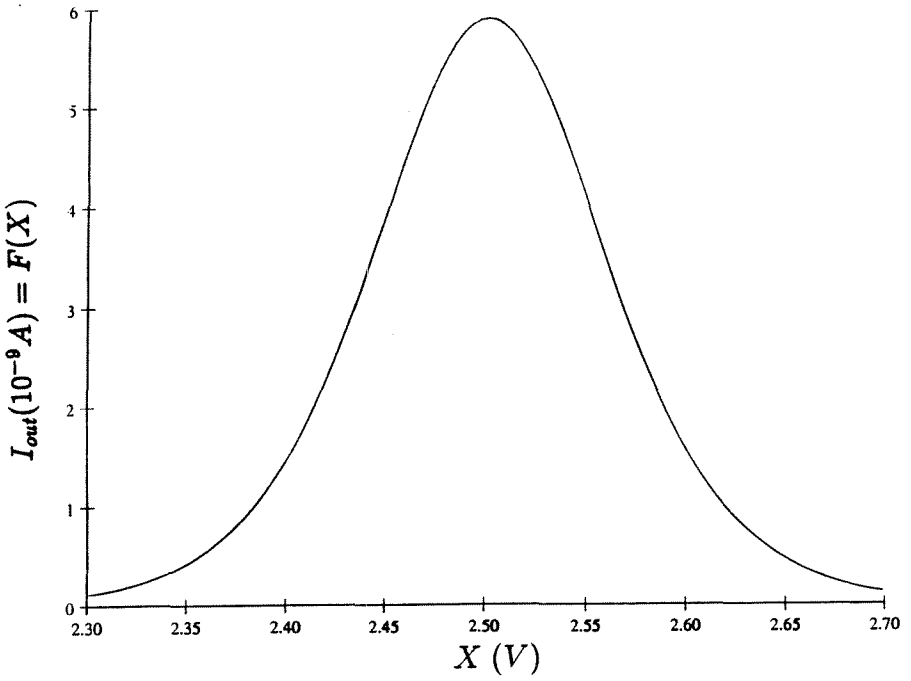

Figure 4.14: Transfer characteristics of a single bump circuit for $X_{ref} = 2.50V$.

Since our circuit internally produces maxima instead of minima, we implemented a gradient ascent operation,

$$\dot{X} \simeq \frac{F(X + \Delta X) - F(X)}{\Delta X} \tag{4.21}$$

instead of gradient descent operation, due to the fact that finding a minimum by gradient descent is equivalent to finding a maximum by gradient ascent. Thus, the circuit of Figure 4.13, starting from an initial point, employs gradient ascent on $F(X)$ to converge to a prelocated maximum. Before discussing this dynamics operation of the circuit, we first consider the implementation of the objective function in greater detail.

The bump circuit cascade (bottom layer of Figure 4.13) has a common input X and a common output I_{out} (i.e., $F(X)$). This enables us to produce a highly nonlinear objective function with three local maxima. More complex functions can be approximated by adding additional bump circuits. Furthermore, the independent control inputs (i.e., separate X_{ref} and V_b) for each bump circuit provide the flexibility to create each maximum at specified location with specified steepness. An example of an objective function, that can be implemented in this circuit, is shown in Figure 4.15.

Figure 4.15 was obtained by sweeping the common input voltage X from 2V to 3V

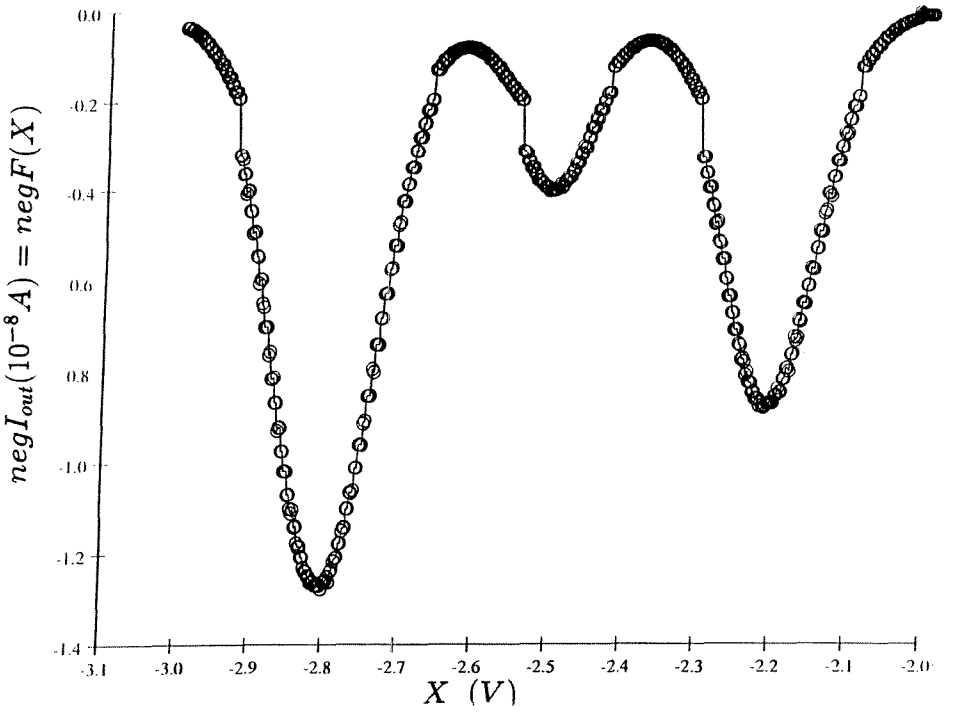

Figure 4.15: Measured objective function with three local minima

and observing the negative value [neg] of the common output current I_{out} (i.e., $negI_{out}$ or $negF(X)$), via a current mirror denoted by CM in Figure 4.13. As expected, plotting the negative value of the output produces three minima, which is not to be confused with the internal gradient ascent operation of the circuit on maxima. The discontinuities at $2nA$ level, seen in Figure 4.15, are due to the calibration errors of the ammeter, which reduces its gain at every decade during the automatic range switching process. The parameters of the circuit required to produce the objective function of Figure 4.15 are given in Table 4.1.

Table 4.1: Parameters for the circuit of Figure 4.13 which produce the objective function of Figure 4.15.

Bump #	X_{ref}	V_b	minimum at
1	$2.2V$	$0.80V$	$X = 2.2V$
2	$2.5V$	$0.75V$	$X = 2.5V$
3	$2.8V$	$0.82V$	$X = 2.8V$

While the bottom layer of the gradient circuit produces $F(X)$ (i.e., $I_{out}(X)$), the top

layer produces $F(X+\Delta X)$ (i.e., $I_{out}[X+\Delta X]$), which is due to an additional transistor Q^* employed only in the top layer of the circuit (see Figure 4.13). The transistor Q^* is placed on the left branch of each upper differential pair and therefore imposes a symmetry-breaking operation. This symmetry-breaking behavior can be analyzed as follows.

We assume that the physical layout of Q^* can have different dimensions (i.e., $I_0^* \sim w^*/l^*$) than other transistors (mostly with $I_0 \sim w/l = 1$) in the circuit, where w and l are the width and length of the transistor gate respectively, and I_0 is the leakage current of transistor Q. Therefore the current flowing through the left branch of each upper differential amplifier is given by

$$I_1^* = (I_0 + I_0^*)\, e^{\kappa X - V^*} \tag{4.22}$$

where, V^* is the common node voltage of the differential pair as shown in Figure 4.13. In order to see the effect of I_0^*, the above equation can be reformulated as,

$$I_1^* = I_0\, e^{\kappa(X-\alpha)-V^*} \tag{4.23}$$

where, α is the explicit contribution of transistor Q^* and can be solved by equating (4.22) and (4.23), which yields

$$\alpha = -\frac{V_T}{\kappa}\, \log \frac{I_0 + I_0^*}{I_0}. \tag{4.24}$$

From equation (4.23) it is clear that the transistor Q^* causes a shift in the X axis by α, whose value is set by (4.24). Hence, as a special case if $I_0^* = I_0$, then $\alpha = -25mV$.

So, while the output of an ordinary voltage correlator bump circuit at the bottom layer of Figure 4.13 yields,

$$I_{out} = \frac{I_1 I_2}{I_1 + I_2} = \frac{I_b}{4}\, \mathrm{sech}^2\, \frac{\kappa(X - X_{ref})}{2} \tag{4.25}$$

the output voltage of the modified voltage correlator bump circuit with the symmetry

breaking transistor Q^* at the top layer gives,

$$I^*_{out} = \frac{I^*_1 I^*_2}{I^*_1 + I^*_2} = \frac{I_b}{4} \, sech^2 \frac{\kappa(X - \alpha - X_{ref})}{2} \tag{4.26}$$

where, $I_1 = I_0 \, e^{\kappa X - V}$ and $I_2 = I_0 \, e^{\kappa X_{ref} - V}$ are the left and right branch currents respectively, I^*_1 is given in (4.23), and $I^*_2 = I_0 \, e^{\kappa X_{ref} - V^*}$. From equations (4.25) and (4.26), it simply follows that

$$I^*_{out}[X] = I_{out}[X + \Delta X] \quad \text{with} \quad \Delta X = -\alpha \, . \tag{4.27}$$

Thus, if $I^*_0 = I_0$ (i.e., $\alpha = -25mV$), equation (4.27) corresponds to producing $F(X + \Delta X)$, where $\Delta X = -\alpha = 25mV$, which is an offset from $F(X)$. Figure 4.16 displays $F(X + \Delta X)$, for $F(X)$ given as in Figure 4.14 and $\Delta X = 25mV$.

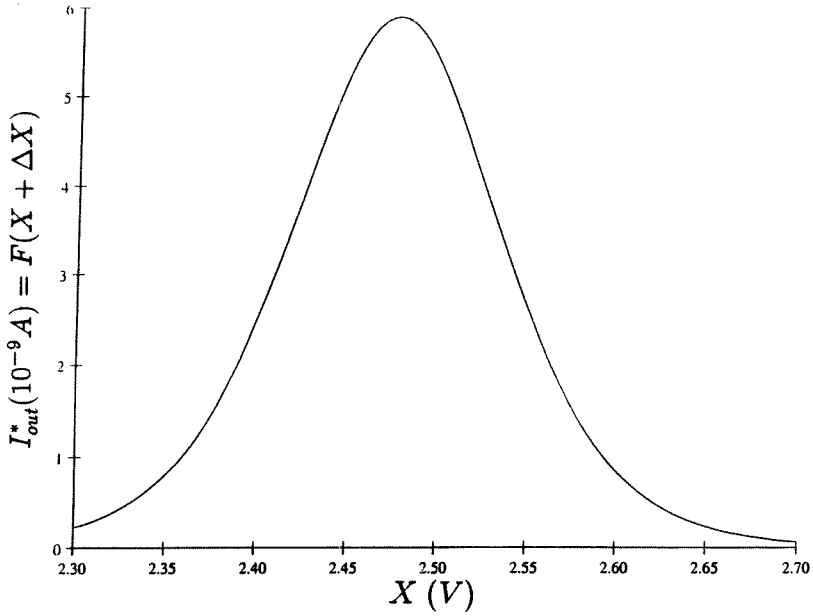

Figure 4.16: The offset function $F(X + \Delta X)$, for $F(X)$ as in Figure 4.14

As Figure 4.14 and Figure 4.16 suggest, each upper bump circuit produces an x-axis shifted (i.e., offset) version of what each lower bump circuit produces. Therefore we can summarize the above findings as follows. Given a state input voltage X, while the bottom

layer generates the output current $F(X)$ the top layer generates the output current $F(X + \Delta X)$, which correspond to the two necessary sample points of the objective function.

Finally, to achieve the gradient ascent system of equation (4.21), the capacitor C in Figure 4.13 integrates the incoming current $F(X) - F(X + \Delta X)$ in the form of the capacitor node voltage,

$$V_C = \frac{1}{C} \int (F(X + \Delta X) - F(X))\, dt \tag{4.28}$$

which is fed back to the circuit, i.e.,

$$X = V_C \, . \tag{4.29}$$

This will advance the sampling points one time-step further. However, it should be kept in mind that while the sampling process is discrete, this advancing process (i.e., integration) is continuous in time. Manipulating equations (4.28) and (4.29) yields,

$$\dot{X} = \lambda \, \frac{F(X + \Delta X) - F(X)}{\Delta X} \tag{4.30}$$

where, $\lambda = \frac{\Delta X}{C}$. Equation (4.30) has the desired form of equation (4.21) and therefore approximates gradient ascent behavior

$$\dot{X} \simeq \lambda \, \frac{dF(X)}{dX} \, . \tag{4.31}$$

Equation (4.31) represents a dynamical system, which has a stable equilibrium point at the local maximum of $F(X)$. The initial condition for the dynamical system is supplied by the variable voltage source X_0, which (similar to the terminal repeller dynamics circuit of Figure 4.8) sets the initial point of X to X_0 when V_{00} is set to 5V such that the transistor Q_i is on and short-circuited. The dynamics of (4.15) is initiated by switching V_{00} from 5V to 0V, which turns off Q_i and thus disconnects the rest of the circuit from X_0.

The effect of the offset, ΔX, on the accuracy and efficiency of the gradient descent circuit is important and must be considered in further detail. From equation (4.27) we

found that $\Delta X = -\alpha$, and combining this with equation (4.24) gives,

$$\Delta X = \frac{V_T}{\kappa} \log \frac{I_0 + I_0^*}{I_0}. \tag{4.32}$$

where, $I_0^* \sim w^*/l^*$. Thus, we have control over the offset ΔX, by changing the width-to-length ratio of the symmetry breaking transistor Q^*.

However, there is a tradeoff between two important factors when choosing the w^*/l^* ratio of Q^*. If w^*/l^* is too large, then ΔX is also large. In this case the gradient ascent approximation (4.30) becomes inaccurate. We thus desire w^*/l^* as small as possible. On the other hand, if w^*/l^* is too small, the gradient circuit may not perform at all due to the mismatches caused by the fabrication. That is, the upper layer of the circuit in Figure 4.13 may match the lower layer, and thus the dynamics of the circuit becomes static.

In order to find an optimal value for w^*/l^* which would accommodate both conditions stated above, we experimented with three different values of w^*/l^* for transistors Q_1^*, Q_2^* and Q_3^* in the layout. Transistors Q_2^* and Q_3^*, with w^*/l^* ratios of $4/50$ and $4/40$ respectively, did not function properly due to the fabrication errors explained above. Q_1^* with $w^*/l^* = 4/8$ resulted in optimal performance.

Testing our gradient ascent chip after fabrication gave the following results. Starting from two initial points at $X_0 = 1.5V$ and $2.6V$ at the basin of attraction of the local minimum at $X = 1.8V$, the dynamics of the circuit converged to the local minimum, as seen in Figure 4.17.

As the figure suggests, for both initial states, the system dynamics converged with no error to the minimum at $X = 1.8V$ when the bias voltage V_{b1} was in the interval $[1.1V \rightarrow 1.2V]$. The subthreshold operation of V_{b1} (i.e., $[0.7V \rightarrow 0.9V]$) degraded the convergence as seen in Figure 4.18, where the solution converged to $X = 1.85V$ with $50mv$ error. This relatively poor convergence was due to the high steepness of the objective function obtained by using a subthreshold V_{b1}, such that $F(X) = F(X + \Delta X)$ for $X = 1.85V$, which assumes an undesired local maximum at $X = 1.85V$ in the approximated gradient dynamics given by equation (4.30). Note that all measured voltages are negative due to the p-well chip,

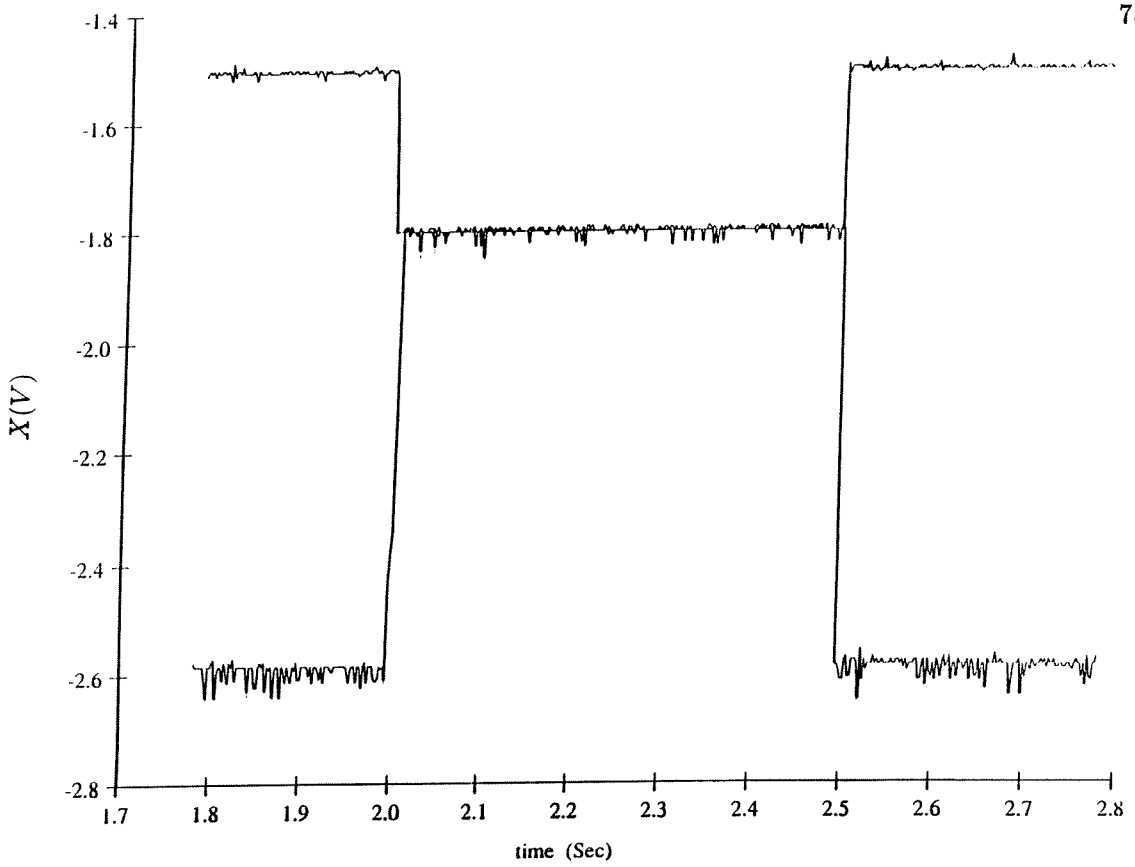

Figure 4.17: Measured time-behavior of voltage X of the gradient ascent chip, for two different initial points.

but for convenience the magnitudes are given here.

From the findings stated above, we conclude that our gradient circuit performs stably and relatively accurately (except for very steep functions) for locating the critical points of arbitrary one-dimensional objective functions. In order to enhance the accuracy of the gradient operation, an alternative method has been proposed by Anderson et al. [AK]. Their model estimates the gradient of the function by using a noise-function correlation approach. Kirk [Kir93] has further extended this approach to handle multiple dimensions. A detailed discussion of this technique can be found in [Kir93].

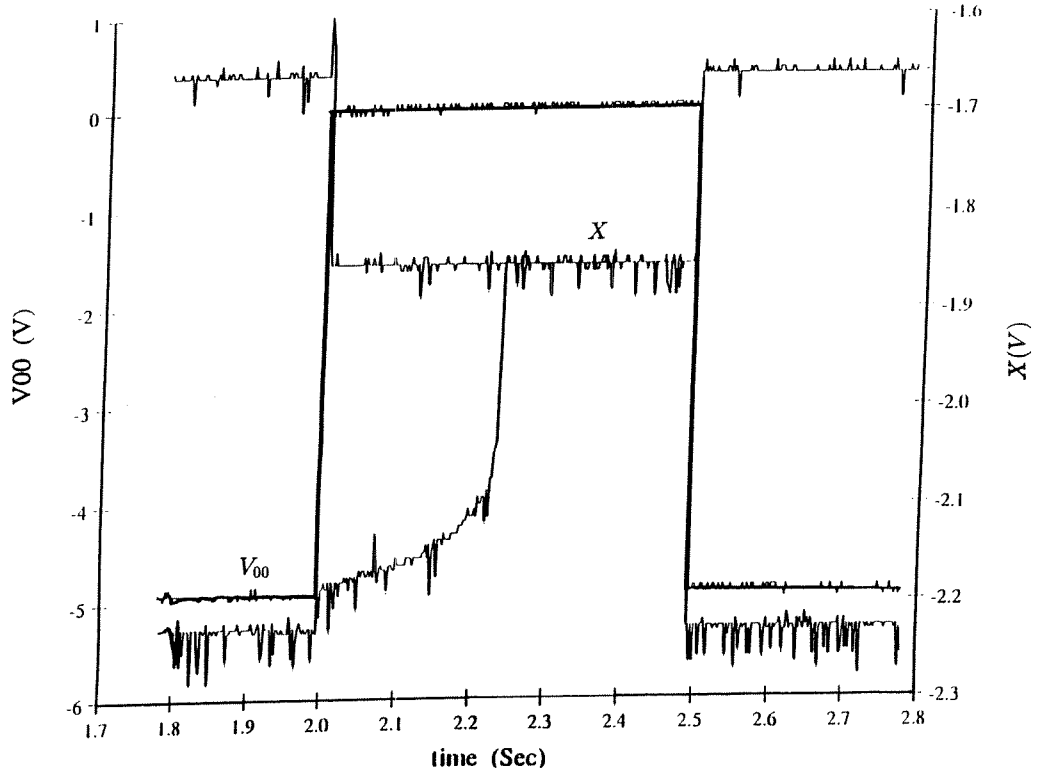

Figure 4.18: Time-convergence of X for subthreshold operation

4.3 Global Optimization Circuit

Here, we show how the concepts mentioned earlier (i.e., Terminal Repeller and Gradient Descent) can be used to develop a global optimization circuit, which additionally requires a control logic circuit. Before describing the modules of the circuit in further detail, let us first consider the theory underlying the behavior of this circuit.

As discussed earlier (Equation (2.28)), the one-dimensional TRUST optimization dynamics consists of the differential equation

$$\dot{X} = -\frac{dF[X]}{dX}\frac{1}{1+\exp(F[X]-F[X^*]+a)} + K\,(X-X^*)^{1/3}\,u(F[X]-F[X^*]) \qquad (4.33)$$

which approximates the following ideal behavior,

$$\dot{X} = \begin{cases} K\,(X-X^*)^{1/3}, & F[X]-F[X^*] \geq 0 \quad \text{repelling phase} \\ -\frac{dF[X]}{dX}, & F[X]-F[X^*] < 0 \quad \text{minimization phase.} \end{cases} \qquad (4.34)$$

This dynamical system locates the global minimum of a given one-dimensional function. However, since our gradient circuit implements gradient ascent, for the purposes of implementation we modify the dynamics in (4.34) as follows:

$$\dot{X} = \begin{cases} K\,(X - X^*)^{1/3}, & F[X] - F[X^*] \leq 0 \quad \text{repelling phase,} \\ \frac{dF[X]}{dX}, & F[X] - F[X^*] > 0 \quad \text{maximization phase.} \end{cases} \tag{4.35}$$

Equation (4.35) performs repelling action whenever the objective function value is below the last found local maximum and, gradient ascent whenever the function value is above the last found local maximum. This guarantees a global ascent behavior (i.e., converging to functionally higher maxima). A schematical representation of the desired dynamics behavior is given in Figure 4.19.

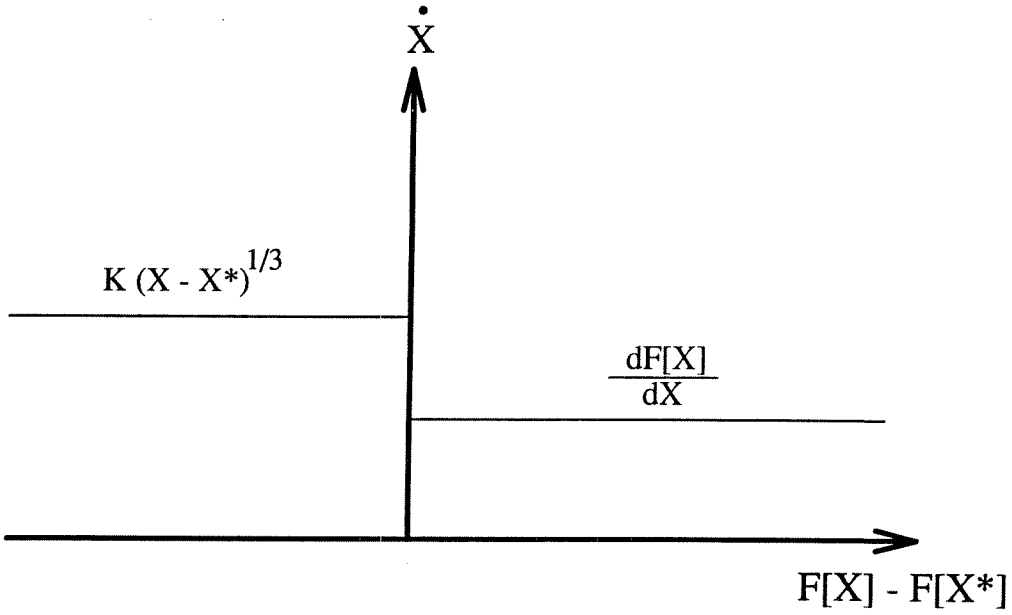

Figure 4.19: Schematical representation for the desired dynamics of the global optimization circuit

The behavior of equation (4.35) shown in Figure 4.19 required designing a control logic circuit, which would switch between the output currents of I_{rep} and I_{gr} according to the condition of the state current $F[X] - F[X^*]$ being negative or positive respectively. Here,

I_{rep} is the repeller output current, which was denoted by I_{out} earlier in equation (4.10), and can be also formulated in the following form using (4.10) and (4.15),

$$I_{rep} = C\,\dot{X} = C\,K\,(X - X^*)^{1/2.43}. \tag{4.36}$$

I_{gr} is the gradient output current, which was previously characterized as $F(X) - F(X + \Delta X)$ in equation (4.28), but can also be shown to be

$$I_{gr} = C\,\dot{X} = F(X) - F(X + \Delta X) \simeq C\,\lambda\,\frac{dF(X)}{dX} \tag{4.37}$$

using equations (4.30) and (4.31).

A winner-take-all circuit was the best candidate to start with for this purpose. Figure 4.20 shows the circuit diagram for the winner-take-all circuit.

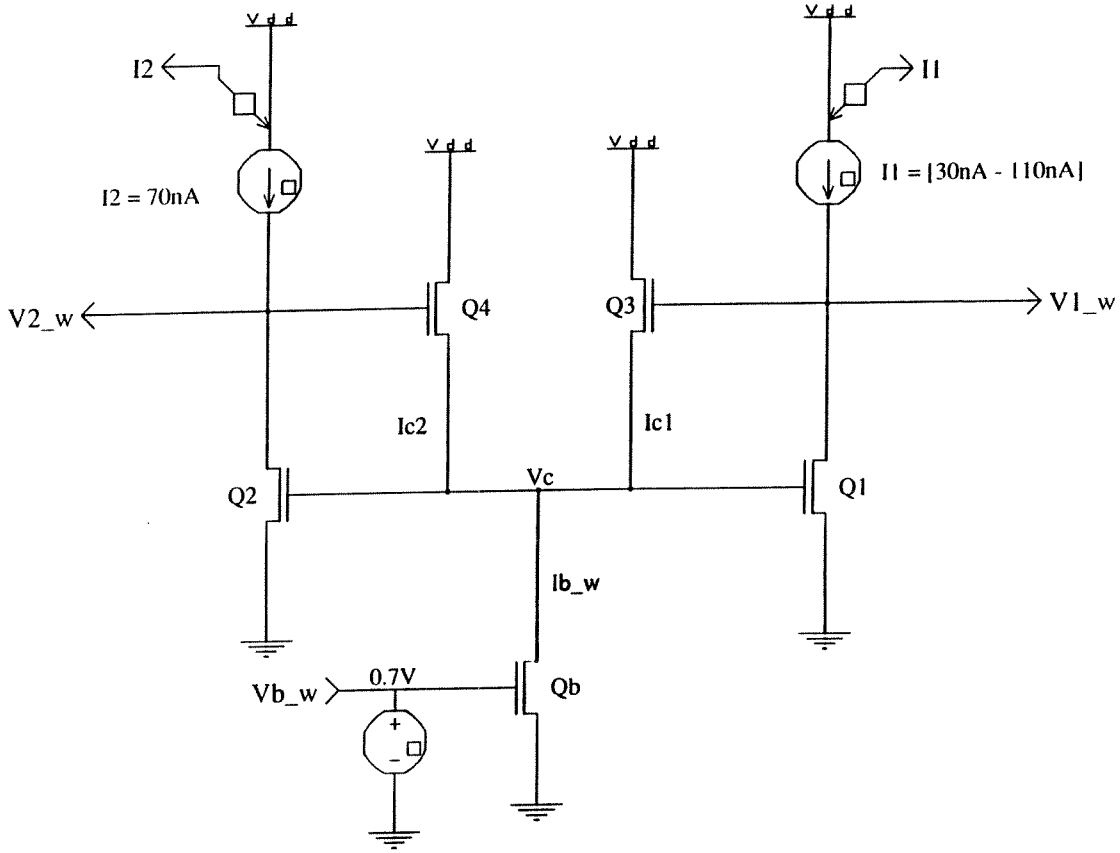

Figure 4.20: Winner take all circuit

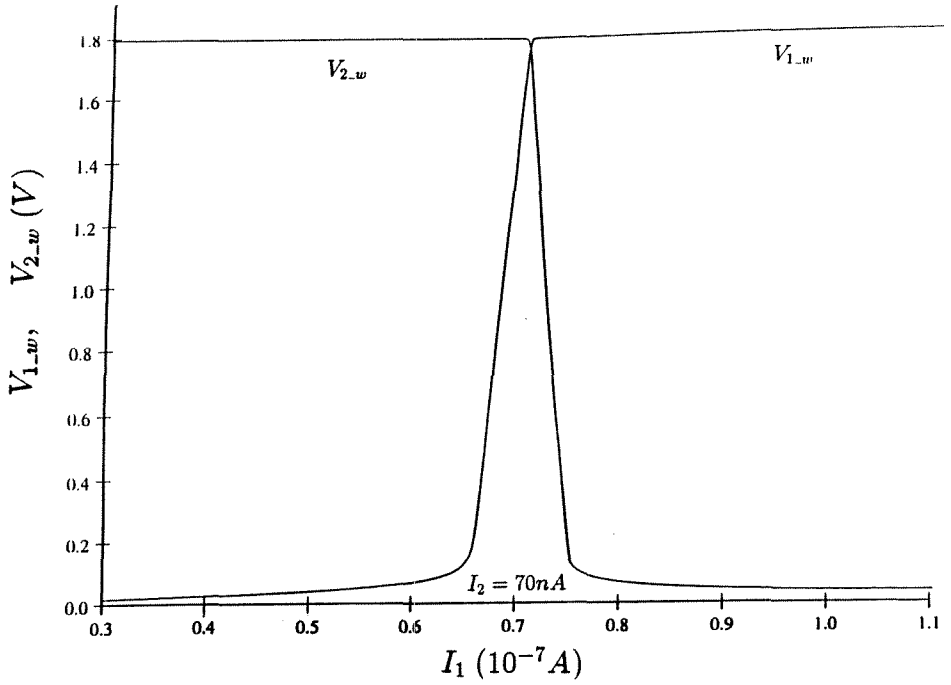

Figure 4.21: Operation characteristics of the winner-take-all circuit

We now discuss the operation of the winner-take-all circuit in detail, as this discussion is necessary to understand the detailed operation of the optimization circuit. The output/input characteristics of the winner-take-all circuit is given in Figure 4.21.

The winner-take-all circuit employs the following behavior. When the input currents I_1 and I_2 are equal, i.e., if we let

$$I_1 = I_2 \equiv I_m, \tag{4.38}$$

the two symmetrical branches of the circuit function in an identical fashion such that for the branch currents the following condition holds.

$$I_{c1} = I_{c2} = I_{b\_w}/2 = (I_0/2)\, e^{\kappa V_{b\_w}}. \tag{4.39}$$

I_{c1} and I_{c2} are the drain currents of the transistors Q_3 and Q_4 respectively, and I_{b_w} is the bias current of Q_b. Due to symmetrical operation, the output voltages will also be identical and can be denoted as

$$V_{1\_w} = V_{2\_w} \equiv V_m. \tag{4.40}$$

Due to equation (4.39), Q_3 and Q_4 will be on and Q_1 and Q_2 will be saturated, such that the input current can be approximated by,

$$I_m = I_0 \, e^{\kappa V_c} \tag{4.41}$$

where V_c is the common node voltage, as seen in Figure 4.20, and can be solved directly from (4.41) to yield,

$$V_c = \frac{V_T}{\kappa} \log \frac{I_m}{I_0}. \tag{4.42}$$

Finally, using the characteristic equations for the branch currents

$$I_{c1} = I_{c2} = I_0 \, e^{\kappa V_m - V_c} \tag{4.43}$$

together with equation (4.39), we can solve for the output voltage V_m

$$V_m = \frac{V_c}{\kappa} + V_{b\_w} - \frac{(\log 2) V_T}{\kappa} \tag{4.44}$$

where V_c is as given in (4.42), and $V_T = 25mV$ is the constant thermal voltage. To summarize, the input current I_m determines the common node voltage V_c, which together with the branch currents and bias current determine the output voltage V_m.

The above discussion concentrated on what happens when the two input currents are equal, such that the circuit is completely balanced. Now we will consider the behavior during imbalance. Assuming the input current I_2 is constant, if the other input current I_1 increases linearly above the value of I_2, since transistor Q_1 will always be more saturated than Q_2, Q_1 will determine the value of V_c, such that V_c will also increase in parallel (actually logarithmically proportional) to I_1, i.e.,

$$I_1 = I_0 \, e^{\kappa V_c}. \tag{4.45}$$

However, since V_c is also a gate voltage to Q_2, an increase in V_c will correspond to a

decrease in the drain voltage of Q_2 (i.e., V_{2_w}), since Q_2 wants to keep the input current I_2 constant, as assumed earlier. Thus as seen in Figure 4.21, as I_1 increases above I_2, V_{2_w} first decreases linearly while Q_2 is saturated. Then Q_2 leaves the saturation region with V_{2_w} first decreasing logarithmically and then linearly again until it hits the lower rail (i.e., $V_{2\_w} \sim 0$), which declares Q_2 as the loser transistor. On the other hand, while V_{2_w} decreases towards zero, since such decrease will turn the transistor Q_4 off, the current I_{c2} will also decrease (exponentially in parallel to V_{2_w}'s linear decrease during Q_2's saturated operation), (i.e., $I_{c2} \rightarrow 0$). Thus most of the bias current I_b starts flowing only via Q_3, i.e.,

$$I_{c1} \rightarrow I_b = I_0\, e^{\kappa V_{b\_w}} \tag{4.46}$$

holds, such that V_{1_w} increases via the characteristic equation of Q_3,

$$I_{c1} = I_0\, e^{\kappa V_{1\_w} - V_c}. \tag{4.47}$$

Note that an exponential increase in I_{c1} causes a linear increase in V_{1_w} (since it is the gate voltage of Q_3) until V_{1_w} hits the upper limit which can be solved by the equations (4.47) and (4.46 with $I_{c1} = I_{b\_w}$) to yield,

$$V_{1\_w} = V_{1\_max} = \frac{V_c}{\kappa} + V_{b\_w} \tag{4.48}$$

where, V_c can be found using (4.45). This declares Q_1 as the winner transistor.

Similarly, when the input current I_1 falls below the constant value of I_2, the above conditions reverse. To summarize, V_c will be determined by I_2, since Q_2 is more saturated than Q_1. However the value for V_c will not change due to constant I_2. As I_1 decreases linearly, V_{1_w} of transistor Q_1 will decrease due to constant gate voltage V_c. The decrease in V_{1_w} will be linear during the saturated operation of Q_1 but as soon as Q_1 becomes unsaturated, V_{1_w} will decrease logarithmically and then linearly again until it hits the lower rail and hence Q_1 becomes the loser. The transistor Q_3 turns off, all the bias current

I_{b_w} starts flowing via Q_4, which forces the output voltage V_{2_w} to increase linearly until it hits the maximum limit given by

$$V_{2\_w} = V_{2\_max} = \frac{V_c}{\kappa} + V_{b\_w} \tag{4.49}$$

where, V_c can be found using

$$I_2 = I_0\, e^{\kappa V_c}. \tag{4.50}$$

To summarize the above findings in a simpler format, if we let

$$I_1 \equiv F(X) \tag{4.51}$$

$$I_2 \equiv F(X^*) \tag{4.52}$$

then for our purposes, the winner-take-all circuit produces the following behavior (for $V_{b\_w} = 0.7V$),

$$\begin{aligned} V_{2\_w} = V_{2\_max} \simeq 1.8V, \quad V_{1\_w} \simeq 0V \quad \text{for } F(X) \ll F(X^*) \\ V_{1\_w} = V_{1\_max} \simeq 1.8V, \quad V_{2\_w} \simeq 0V \quad \text{for } F(X) \gg F(X^*)\ . \end{aligned} \tag{4.53}$$

Although (4.53) implies the desired switching operation of equation (4.35), the outputs of the winner-take-all circuit are voltages (i.e., V_{2_w} and V_{1_w}), but not currents (i.e., desired to have I_{rep} and I_{gr} as given in (4.36) and (4.37)). Therefore, we wanted to have a circuit that would implement the following behavior,

$$I_{out} = \begin{cases} I_{rep}, & \text{for } F(X) < F(X^*) \quad \text{i.e., repelling phase} \\ I_{gr}, & \text{for } F(X) > F(X^*) \quad \text{i.e., maximization phase}\ . \end{cases} \tag{4.54}$$

To satisfy the above criterion, we designed a modified winner-take-all circuit with current-mode-operation as seen in Figure 4.22.

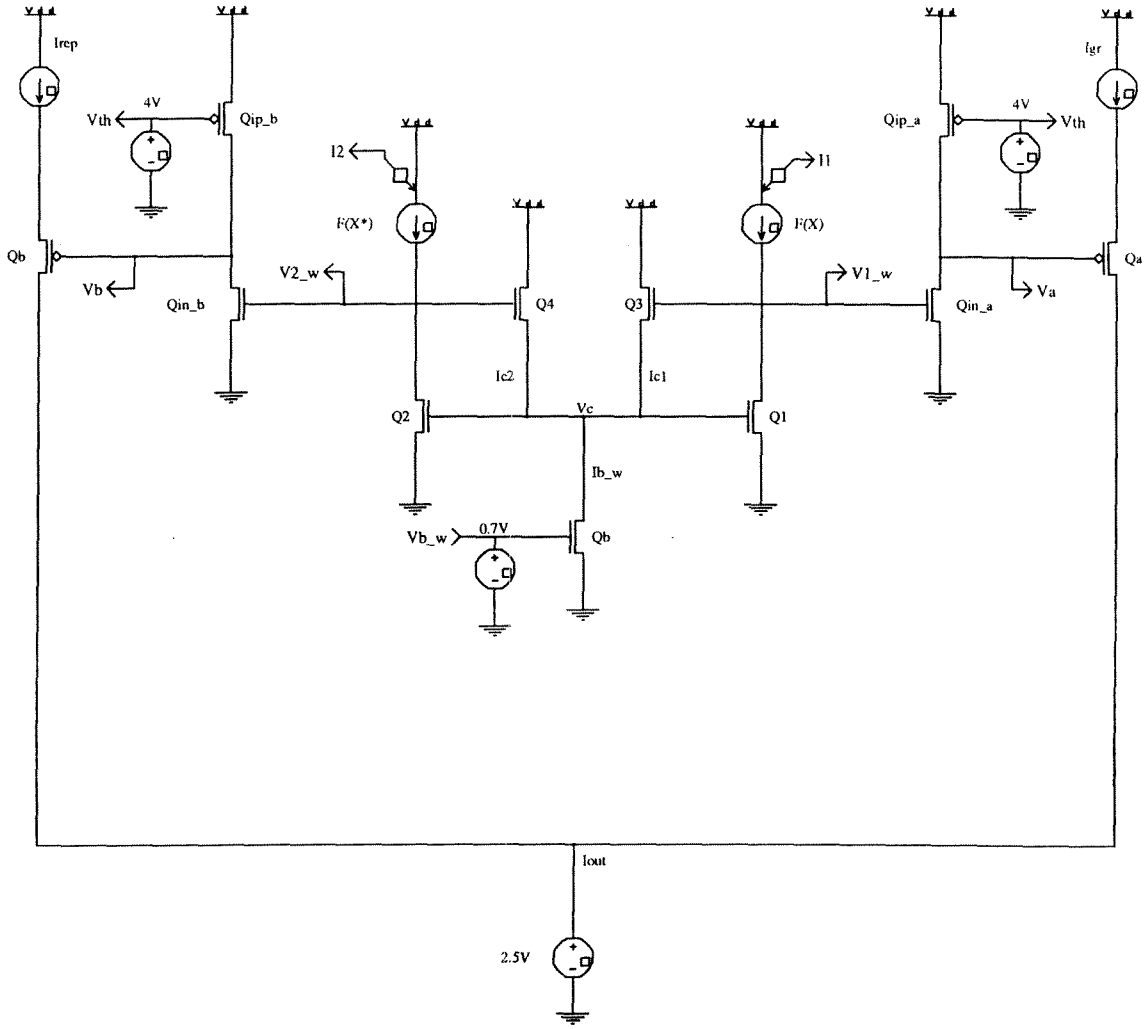

Figure 4.22: Modified winner-take-all circuit with current-mode-operation

The standard winner-take-all of Figure 4.20 can be seen as the center component of this new circuit as well. Additionally, note that on the right-hand side of the circuit, the node denoted by V_{1_w} (which is one of the output voltages of the standard circuit) is connected to an inverter consisting of the transistors Q_{in_a} and Q_{ip_a} and a voltage source V_{th}. V_{th} is set to $4V$ and imposes a threshold to the inverting operation. Thus, if transistor Q_1 is a winner, such that $V_{1\_w} = V_{1\_max} \simeq 1.8V$ holds, the magnitude of the gate to source voltage of Q_{in_a}

(i.e., $|V_{gs\_in}| \simeq 1.8V$) becomes larger than the magnitude of the gate to source voltage of Q_{ip_a} (i.e., $|V_{gs\_ip}| = 5 - V_{th} = 1V$). However, since both transistors Q_{in_a} and Q_{ip_a} share the same channel current, the above condition forces Q_{in_a} to become unsaturated with voltage V_a tending towards zero. Also note that the node denoted by V_a is connected to p-channel transistor Q_a, which acts like an inverted switch, i.e., turns on when $V_a \simeq 0V$, and lets the I_{gr} current flow freely.

On the other hand if Q_1 is the loser, all the above conditions reverse in effect. Initially V_{1_w} converges to zero, so that this time $|V_{gs\_in}| \simeq 0V < |V_{gs\_ip}| = 1V$ holds, which drives Q_{ip_a} to become unsaturated and V_a to the upper rail (i.e., $V_a \simeq 5V$). Consequently, this blocks the I_{gr} current (i.e., $I_{gr} \simeq 0A$) by turning off the Q_a switch.

The above process describes the behavior of the right side of the circuit. Since the circuit (Figure 4.22) is symmetrical in structure (but not necessarily in function), the left side of the circuit performs an action which is similar but opposite to the behavior of the right side of the circuit described above, i.e., when the transistor Q_1 is a winner and Q_2 is a loser, the current I_{rep} is blocked; and when Q_1 is a loser but Q_2 is a winner, the current I_{rep} is allowed to pass freely.

Under the light of the above explanations the operation of the modified winner-take-all circuit can be summarized as

$$I_{out} = \begin{cases} I_{rep}, & \text{when } Q_2 \text{ is a winner and } Q_1 \text{ is a loser} \\ I_{gr}, & \text{when } Q_1 \text{ is a winner and } Q_2 \text{ is a loser .} \end{cases} \tag{4.55}$$

For a more detailed understanding of the circuit, we did the following simulations, analysis and deductions.

Figures 4.23 and 4.24 displays a simulation that demonstrates the operation of the modified winner-take-all circuit. The external current sources I_{rep} and I_{gr} are set to fixed values of $300nA$ and $100nA$ respectively. The winner-take-all operation is tested by setting $F(X^*)$ to a constant value of $70nA$ and sweeping $F(X)$ from 58 to $78nA$ (Figure 4.23).

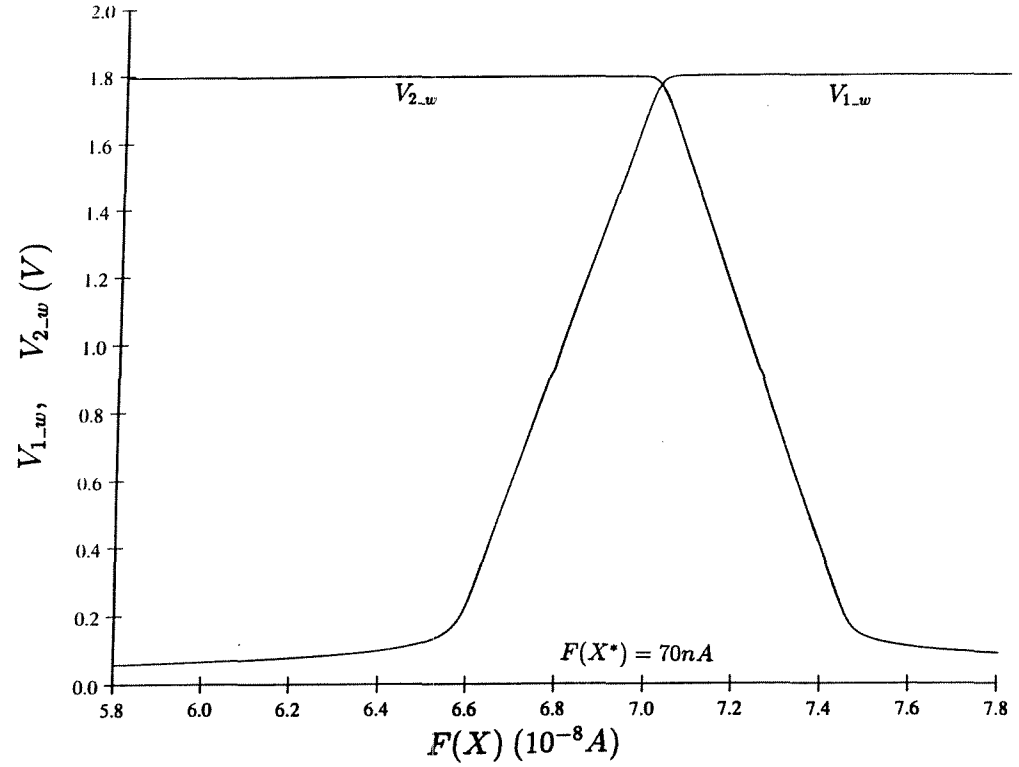

Figure 4.23: V_{1_w} vs. V_{2_w} characteristics for the modified winner-take-all circuit

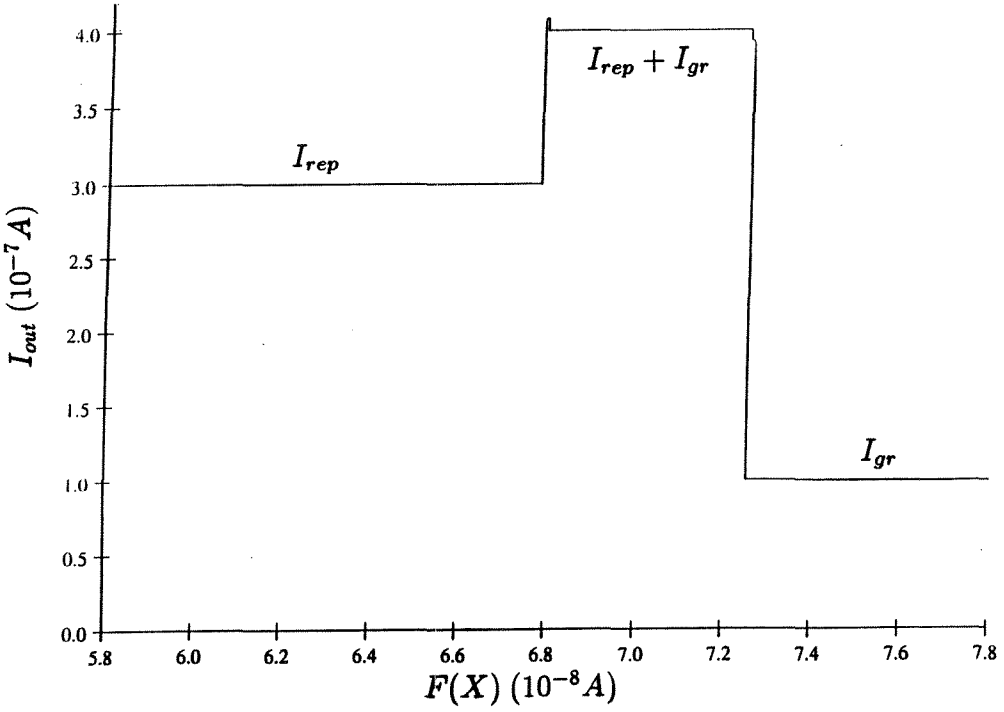

Figure 4.24: The output characteristics of the modified winner-take-all circuit for the simulation of Figure 4.23

As Figure 4.24 suggests, when $F(X)$ is much smaller than $F(X^*)$, V_{1_w} is a loser and V_{2_w} is a winner and consequently Q_a is off and Q_b is on such that $I_{out} = I_{rep}$. When F(X) starts to increase (due to external sweeping process), Q_1 becomes saturated and V_{1_w} increases as expected. However, there is a critical point, as soon as V_{1_w} slightly exceeds $1V$ (i.e., $5 - V_{th}$), where it turns Q_a on. Since Q_b is still on, current from both branches flow at the same time, i.e. $I_{out} = I_{rep} + I_{gr}$. This condition lasts as long as both V_{1_w} and V_{2_w} are above the $1V$ level (since that will keep both Q_a and Q_b on) and until F(X) increases to a level where it is somewhat larger than $F(X^*)$, such as to force Q_2 to reduce it's voltage V_{2_w} just below the $1V$ level (while saturated). Hence, Q_b turns off. After this critical condition is over, further increase in $F(X)$ will push Q_2 to fall below saturation, such that V_{2_w} is a loser and V_{1_w} is a winner and Q_b is off and Q_a is on such that $I_{out} = I_{gr}$.

To summarize, the modified winner-take-all circuit performs (i.e., a detailed analysis)

$$I_{out} = \begin{cases} I_{rep}, & \text{for } F(X) < F(X^*) - \epsilon \\ I_{rep} + I_{gr} & \text{for } F(X^*) - \epsilon < F(X) < F(X^*) + \epsilon \\ I_{gr}, & \text{for } F(X) > F(X^*) + \epsilon \end{cases} \tag{4.56}$$

where, $\epsilon = |F(X^*) - F(X)_\epsilon|$, and $F(X)_\epsilon$ is the value of the current $F(X)$ at the point where the output voltage V_{1_w} or V_{2_w} intersects with the 1V (i.e., $5 - V_{th}$) value level (see Figure 4.23).

The behavior of the modified winner-take-all circuit (Figure 4.22) in equation (4.56) is very close to the desired global ascent behavior of (4.54), with the exception of the second line of equation (4.56), which implies that both currents (i.e., I_{rep} and I_{gr}) flow at the same time during a small period of time (as seen in Figure 4.24). This drawback in the operation of the circuit is due to the fact that there does not exist a sharp transition from the first phase (i.e., Q_2 is a winner and Q_1 a loser) to the second phase (i.e., Q_1 is a winner and Q_2 a loser), but the transition is a smooth one, which creates a small region, where there exist no distinct winner or loser. Instead both transistors Q_1 and Q_2 are saturated. During part of this region, especially when both of the drain voltages of Q_1 and Q_2 (i.e., V_{1_w} and V_{2_w})

are above $1V$ value, both Q_{in_a} and Q_{in_b} are forced to fall below saturation with V_a and V_b at the lower rail (i.e., $0V$). This arises because the magnitude of the gate-to-source voltage of Q_{in_a} and Q_{in_b} (which are V_{1_w} and V_{2_w}) is larger than the magnitude of the gate-to-source voltage of Q_{ip_a} and Q_{ip_b} (each of which is equal to $5 - V_{th} = 1V$) respectively. Therefore both switches Q_a and Q_b are on during this period which enable both currents I_{rep} and I_{gr} to pass freely, leading to the undesired situation mentioned above.

As we shall subsequently see, this undesired condition does not pose a problem for the overall performance of the global optimization process. However, a remedy will also be presented to completely eliminate the problem.

With this review of the major components in mind, we now consider the overall behavior of the global optimization circuit. Figure 4.25 shows a complete global optimization circuit, consisting of the terminal repeller circuit, the gradient descent circuit, and the modified winner-take-all circuit.

On the far right side of Figure 4.25, the gradient ascent circuit can be seen. Two cascade modules (i.e., voltage correlator bump circuits) internally produce the test objective function on which the global maximum will be searched. The test function consists of two local maxima, one of which is global, as can be seen in Figure 4.26. Note that this function is plotted by sweeping the input voltage X (i.e., input to the gradient ascent circuit) and measuring the negative value of the current $F(X)$ (via the node denoted by $negF(X)$) as the y-axis. Hence the objective function is inverted, with maxima appearing as minima. The test function has the qualitative characteristics shown in Table 4.2.

Table 4.2: Parameters for the objective function shown in Figure 4.26

Max #	X	$F(X)$	V_b
1	$2.2V$	$28.281nA$	0.76
2	$2.8V$	$72.114nA$	0.80

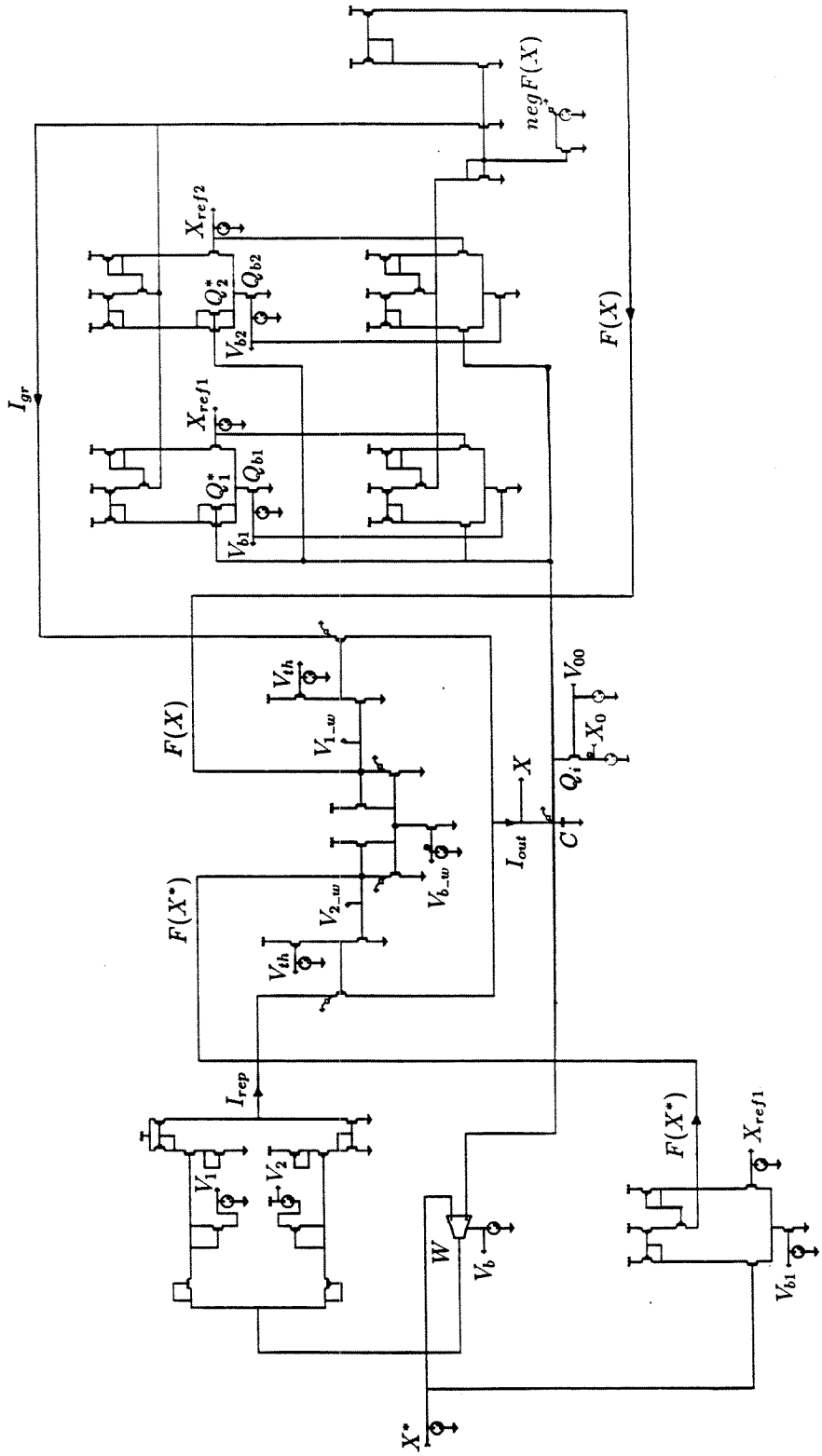

Figure 4.25: Global optimization circuit

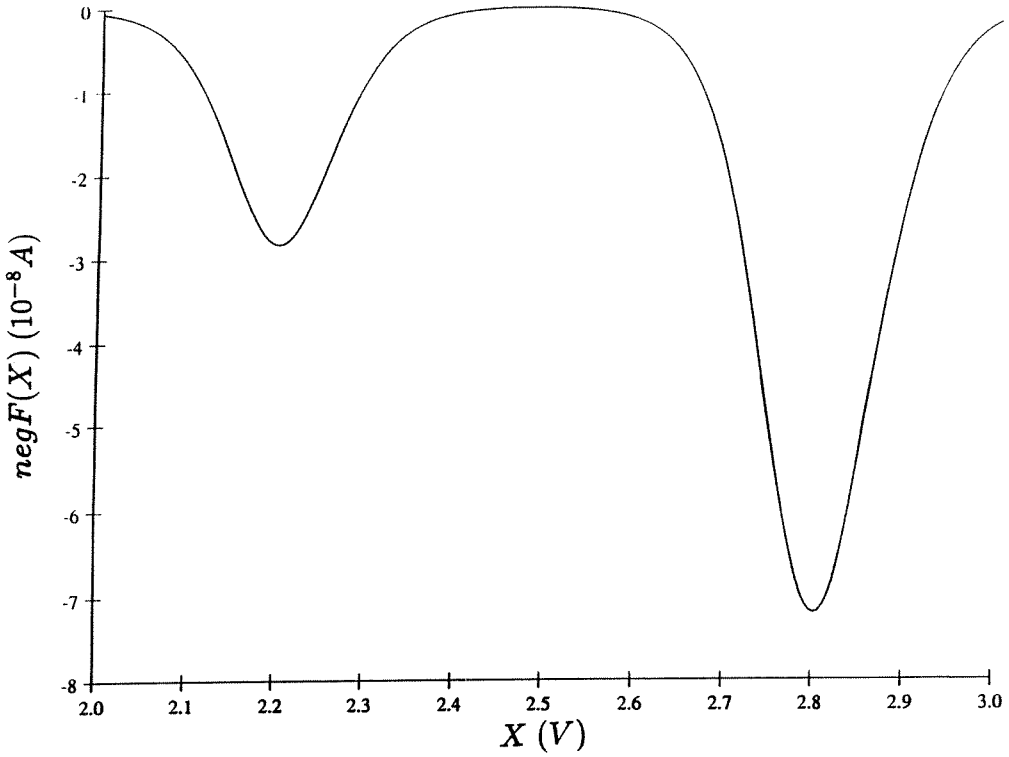

Figure 4.26: The test function for global optimization

The width-to-length ratio of the symmetry-braking transistors Q_1^* and Q_2^* in the upper layer of the circuit (see the gradient ascent block of Figure 4.25) is adjusted to the optimal value of 4/8 (based on the previous findings of page 74) to select the offset ΔX, which effects the efficiency of the gradient ascent operation (recall (4.30), (4.32) and (4.37)).

Before checking the global convergence, the local convergence of the gradient ascent part of the global circuit was checked for the two local maxima of Figure 4.26. When this circuit was initiated at $X_0 = 2V$, which was in the basin of attraction of the local maximum at $X = 2.2V$, the gradient ascent circuit converged to $2.177V$ with $23mV$ error. On the other hand, when it was initiated at $X_0 = 2.6V$ (in the basin of attraction of the local maximum at $X = 2.8V$), it converged to $2.769V$ with $31mv$ error. Both results confirmed the operational efficiency of the gradient ascent part of the global circuit on the given test function. The errors, which were expected due to the high steepness of the objective function during subthreshold operation of Q_b's, were minimal. A simulation of the local convergence for one of the cases above can be observed in Figure 4.27.

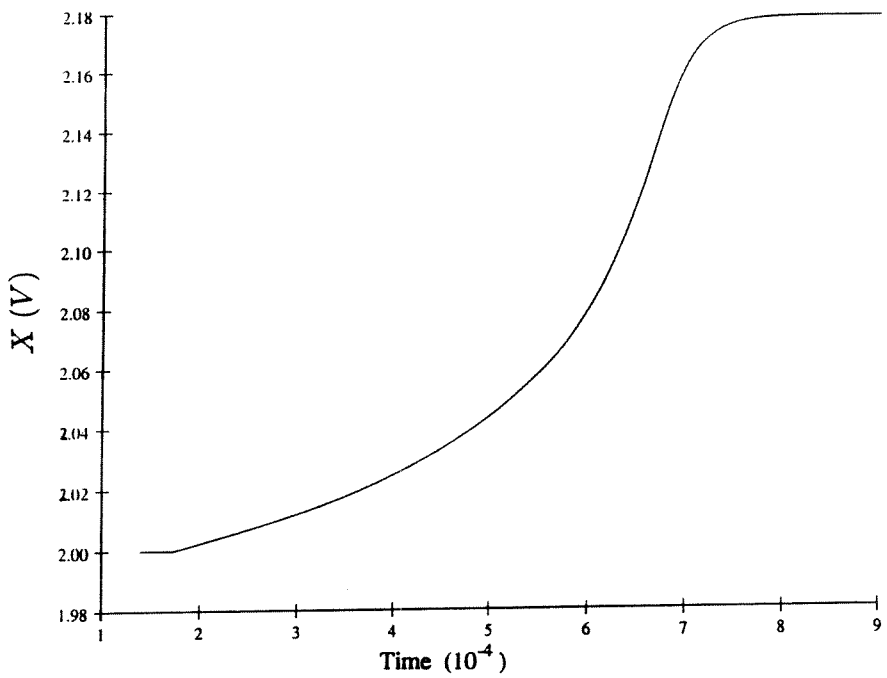

Figure 4.27: Local convergence of the gradient ascent part of the global circuit to the maximum located at $X = 2.2V$ for the test function of Figure 4.26.

After having checked the efficiency of the gradient ascent circuit on the given objective function of Figure 4.26, we continue analyzing the global optimization circuit of Figure 4.25. Note that the gradient ascent block receives the capacitor node voltage X (also the state variable) as the input and produces two different outputs, current $F(X)$ (i.e., function value at X) and the gradient current I_{gr} of equation (4.37).

The terminal repeller circuit be seen on the very left side of Figure 4.25. This circuit receives the same input X, as well as another fixed input X^*. The value of X^*, which can be adjusted by an external voltage source, determines the location of the terminal repeller, which was emphasized in equations (4.15) and (4.35). The terminal repeller block produces the output current I_{rep} as given in (4.36).

On the bottom left corner of the global optimization circuit, there is a single voltage correlator bump circuit, which outputs the fixed function value at the terminal repeller point, i.e., $F(X^*)$, given X^* as the input.

Finally, the modified winner-take-all circuit is located at the center of Figure 4.25. It receives the repeller current I_{rep} from the terminal repeller circuit, the gradient current I_{gr}

from the gradient ascent circuit and two other input currents, $F(X)$ and $F(X^*)$, which are the decision components of the circuit required for the necessary switching operation between the main components of I_{rep} and I_{gr}. The modified winner-take-all circuit produces the output current I_{out}, as given in equation 4.56 and repeated below.

$$I_{out} = \begin{cases} I_{rep}, & \text{for } F(X) < F(X^*) - \epsilon \\ I_{rep} + I_{gr} & \text{for } F(X^*) - \epsilon < F(X) < F(X^*) + \epsilon \\ I_{gr}, & \text{for } F(X) > F(X^*) + \epsilon \ . \end{cases} \tag{4.57}$$

The output current I_{out} is integrated at the terminals of the capacitor C to produce the state voltage X according to

$$I_{out} = C \ dX/dt. \tag{4.58}$$

Consequently, X is fed back to the inputs of both the terminal repeller circuit and gradient ascent circuit (see Figure 4.25) in order to complete the global optimization dynamics.

To summarize, the global circuit implements the following dynamics,

$$\dot{X} = \begin{cases} K\,(X - X^*)^{1/3}, & \text{for } F(X) < F(X^*) - \epsilon \\ K\,(X - X^*)^{1/3} + \lambda \frac{dF[X]}{dX} & \text{for } F(X^*) - \epsilon < F(X) < F(X^*) + \epsilon \\ \lambda \frac{dF[X]}{dX}, & \text{for } F(X) > F(X^*) + \epsilon \ . \end{cases} \tag{4.59}$$

Based on these dynamics, as will be seen in the following simulations, the global circuit has a "global ascent" property, and therefore globally optimizes the given one-dimensional objective function $F(X)$. It should be noted that equation (4.59) is very close to the originally desired dynamics of (4.35). Additionally, the ensuing simulation results demonstrate that the performance of (4.59) is equivalent to the optimal dynamics of (4.35), under a broad region of the circuit parameters. Furthermore, after presenting the simulations, a technique will be introduced to bring ϵ of equation (4.59) to zero such that the desired form of (4.35) will be achieved via the global circuit.

The dynamics of the global circuit (Figure 4.25) is initiated by the voltage source denoted by X_0, which keeps the capacitor voltage X constant at the initial level X_0 (i.e., $X = X_0$)

for time $t < 0$. This effect is due to the transistor Q_i which is shorted when V_{00} is set to $5V$. Switching V_{00} to $0V$ turns Q_i off and disconnects voltage source X_0 from the rest of the circuit, thereby initiating the dynamics of (4.59) at $t = 0$.

Next we discuss the simulated performance of the global optimization circuit on the internally produced test function of Figure 4.26. The idea behind the simulation was to demonstrate the global ascent property of the circuit, by letting the circuit locate the global maximum of the objective function after being initiated with a small perturbation from the local maximum at $X = 2.2V$. Adjusting the necessary circuit parameters before the simulation was of primary importance. We set the voltage source X^* to $2.2V$, thereby placing a terminal repeller at the predetermined local maximum. The starting point for the optimization was established by setting the voltage source X_0 to 2.231 value. Additional parameters were the gain factors and the bias voltage of the terminal repeller circuit which were adjusted as follows: $V_1 = 0.50V$, $V_2 = 0.54V$ and $V_b = 0.85V$. These parameters, as discussed earlier on page 63, determined the power of the repeller. Finally, the bias voltage V_{b_w} of the modified winner-take-all circuit was set to $1V$ for proper biasing and both threshold voltages V_{th} were fixed to $4V$.

Figures 4.28 – 4.33 show the results of a simulation (denoted by S1).

As mentioned earlier, the simulation is performed on the objective function $F(X)$, whose inverted value was shown in Figure 4.26. In order to demonstrate the global ascent property of the global optimization circuit of Figure 4.25, we chose the domain of interest for optimization to be the region displayed by Figure 4.26 (i.e., $2.0V < X < 3.0V$), and assumed that the position of the local maximum is predetermined and known (i.e., $X = 2.2V$). That is, we assumed that the local maximum had been found by a previous maximization process. The circuit simulation was initiated with a small perturbation from the local maximum at $X = X_0 = 2.231$ so as to perturb the system into the domain of interest via right flow. Hence the circuit was expected to escape the basin of attraction of the local maxima and converge to global maximum of the test function via subsequent repelling and gradient ascent properties of the TRUST dynamics of equation (4.35). Although the circuit at this

level exhibited the approximate dynamics of (4.59), the claim was that this dynamics did not differ from the ideal (4.35) under a broad range of parameters.

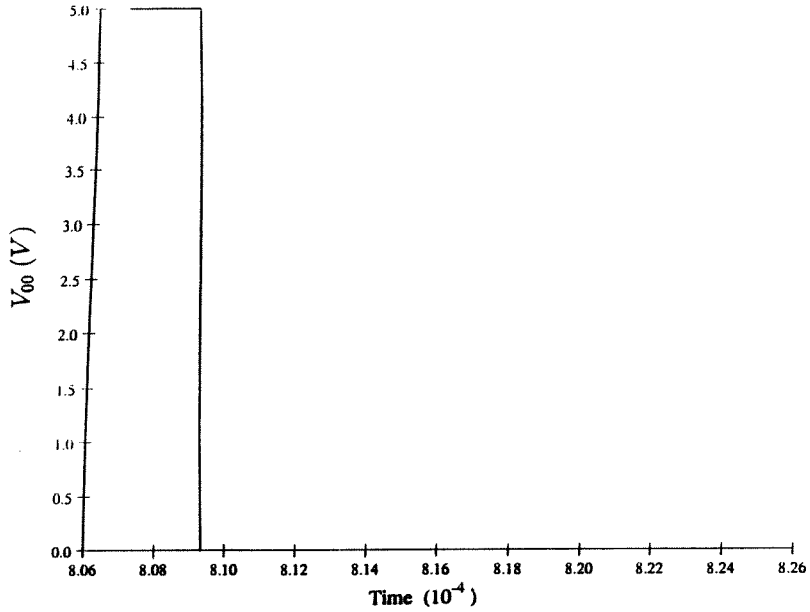

Figure 4.28: The variation of the switch V_{00} vs. time

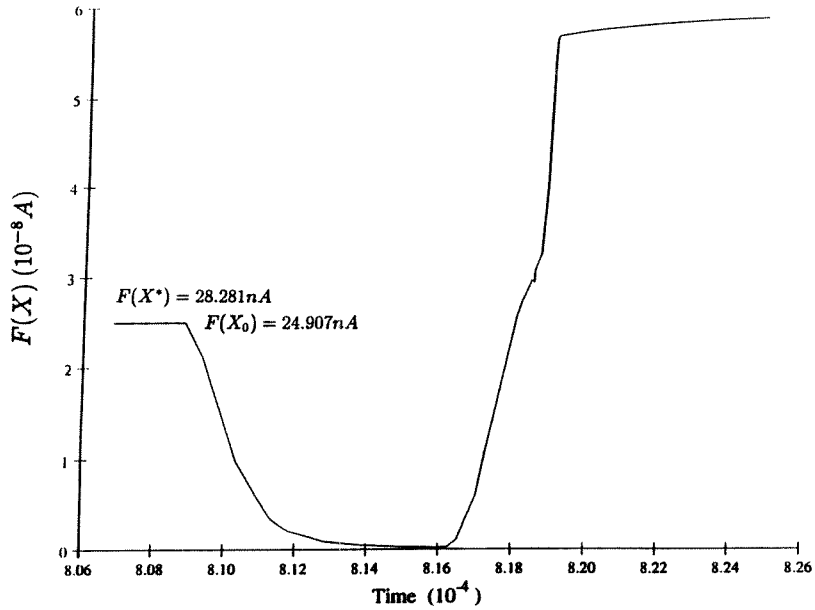

Figure 4.29: The variation of the function value $F(X)$ vs. time for simulation S1 (fixed $F(X^*)$ is also shown).

The simulation is initiated at $t = 0$ by switching V_{00} from $5V$ to $0V$. Figure 4.28 shows this step-like behavior of the V_{00} switch. Note that although the x-axis of most of the graphs displayed here show the actual time, the word "time" will be used to denote the reference time, which has its origin (i.e., $t = 0$) at the instant when $V_{00} = 0V$ (i.e., when the dynamics of the circuit becomes active).

Figure 4.29 displays the variation of the function value $F(X)$ vs. time. Note that for $t < 0$, F(X) is constant at $F(X_0) = 24.907nA$, since during this time the state voltage X is set to $X_0 = 2.231V$. The functional value of the the terminal repeller X^*, placed at the local maximum $X = 2.2V$, can also be seen in Figure 4.29 as $F(X) = F(X^*) = 28.281nA$. At $t = 0$, since $F(X)$ (i.e.,$F(X_0)$) is smaller than $F(X^*)$, the circuit enters the repelling phase as discussed earlier and in accordance with the first line of the equations (4.57) and (4.59). The activation of the repeller current I_{rep} during this phase can be seen in Figure 4.30, which shows the time variation of the repeller current I_{rep}. The winner-take-all circuit, while letting the repeller current I_{rep} freely flow, blocks the gradient current I_{gr}. This can be seen as zero I_{gr} current in Figure 4.31, which presents the time variation of I_{gr}. The output current, I_{out}, which is integrated at the terminals of the capacitor C, therefore is equivalent only to the repeller current I_{rep} during the repelling phase, as Figure 4.32 suggests.

The repeller is active and repels the state X (via positive flow) for all the region of the objective function that lies below the initial local maximum (i.e., for $F(X) < F(X^*)$). The time evolution of the state voltage X is displayed in Figure 4.33. Note how the dynamics of the circuit due to the repeller pushes the state X to increase during this phase. The corresponding initial decrease and subsequent increase in the functional value of $F(X)$ (Figure 4.29) demonstrates that the circuit manages to escape the basin of attraction of the local maximum and enters the basin of the global maximum during the repelling phase. It is also important to remark that the repeller current I_{rep} (which is a measure of the velocity of the repelling state) does not increase continuously (Figure 4.30) during this phase, but saturates by an upper limit, despite the fact that a real terminal repeller is expected to

increase in velocity, as X moves further away from X^* (see equation (4.59)).

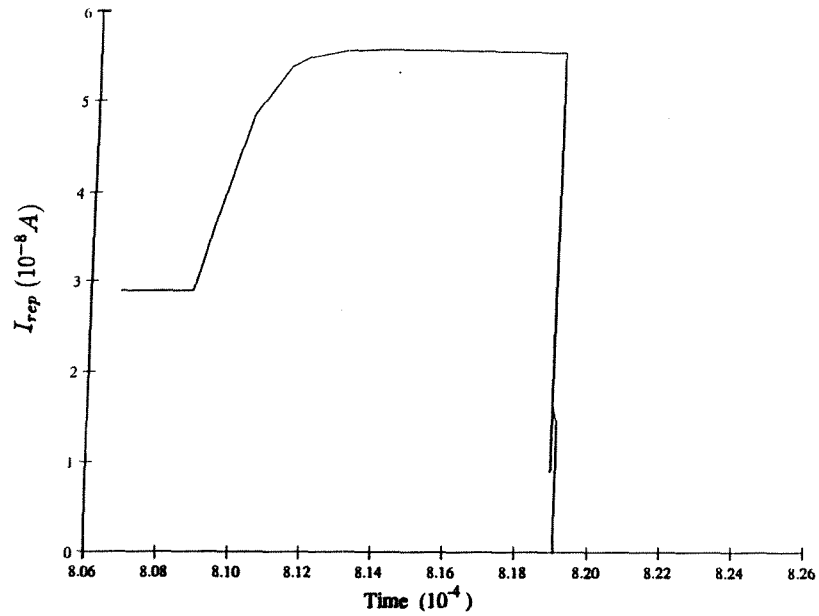

Figure 4.30: The time variation of the repeller current I_{rep} for simulation S1

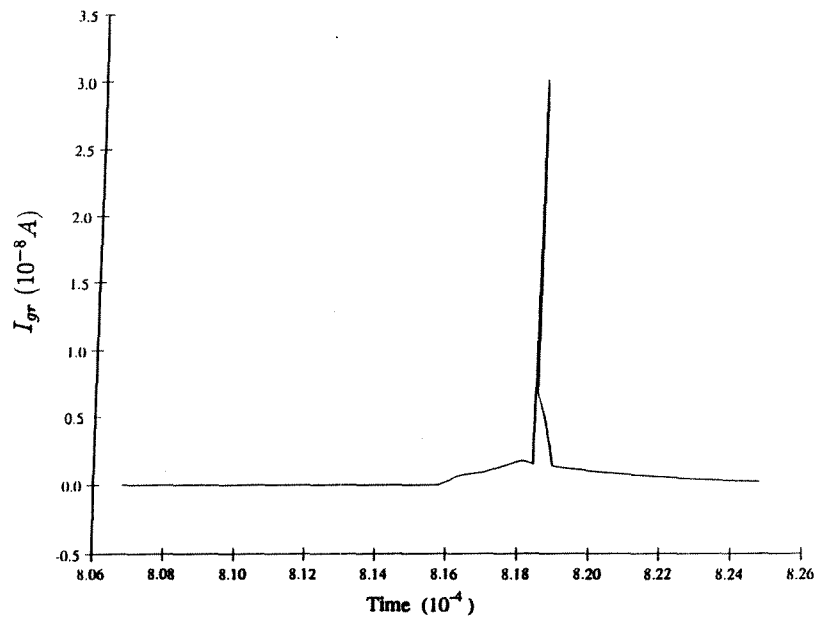

Figure 4.31: The time variation of the gradient current I_{gr} for simulation S1

This behavior, which is due to the saturated nonlinear operation (for the region, where $|X - X^*| \tilde{>} 150mV$) of the wide-range amplifier employed in the terminal repeller circuit, however, does not influence the normal repelling operation, since the state X will be repelled as long as there exist a positive current I_{rep}.

As discussed earlier, when X tunnels to the basin of attraction of the global maximum, such that $F(X)$ becomes equal to $F(X^*)$ with $X \neq X^*$ (i.e., the point in the basin of global maximum, which has the same functional value as the initial local maximum), the transition from the repelling phase to the gradient phase is not sharp, as emphasized in the second line of the equations (4.57) and (4.59).

For the region that corresponds to $F(X^*) - \epsilon < F(X) < F(X^*) + \epsilon$, the winner-take-all circuit allows both currents I_{rep} (Figure 4.30) and I_{gr} (Figure 4.31) to pass freely, such that the output current I_{out} (Figure 4.32) is a summation of both currents. Thus, both the gradient ascent and the repeller effect are imposed on the state voltage X for this small region, and although the gradient ascent is desired to converge to the global maximum in a stable manner, the yet unavoidable repelling effect is redundant and can cause instability or avoidance of the global maximum if the repeller power K is too high. However, this region, where both I_{gr} and I_{rep} is active, is considerably small in time, and therefore if K is chosen low enough, the repeller repels X in the direction of the gradient ascent and merely enhances the convergence performed by the gradient ascent. Additionally, due to low K, the repeller current I_{rep} is also small enough not to carry the state X beyond the global maximum, and as soon as the system enters the region of $F(X) > F(X^*) + \epsilon$, the repeller current I_{rep} becomes inactive such that the normal gradient ascent operation can take over while the state of the circuit is still approaching the global maximum. Note that in parallel with the above discussion, the repeller power K, as determined by the parameters V_1, V_2 and V_b (set to 0.50V, 0.54V and 0.85V respectively for the current simulation), was low enough, and did not in any way negatively influence the normal global optimization operation.

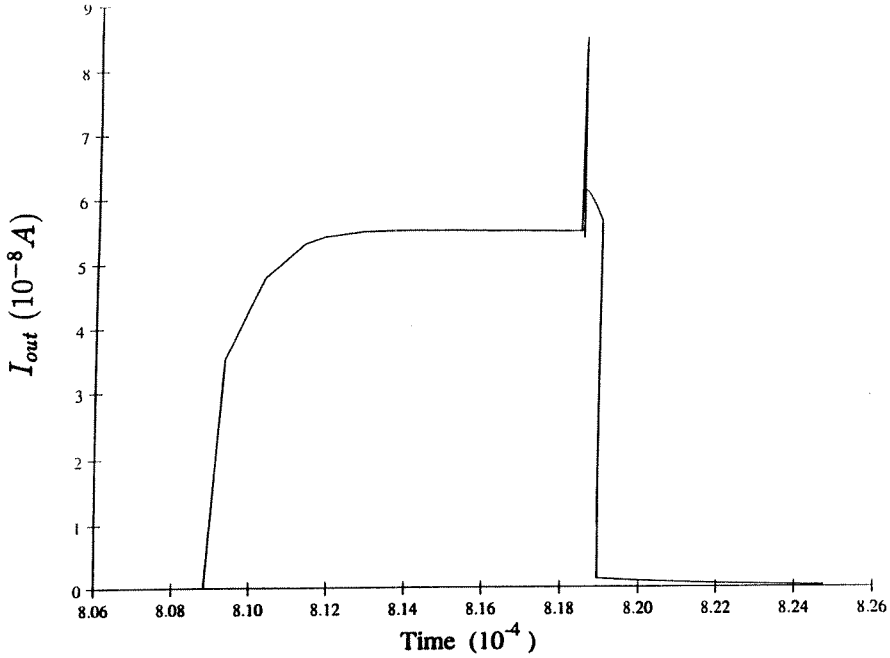

Figure 4.32: The time evolution of the output current I_{out} as given in equation 4.57 for simulation S1.

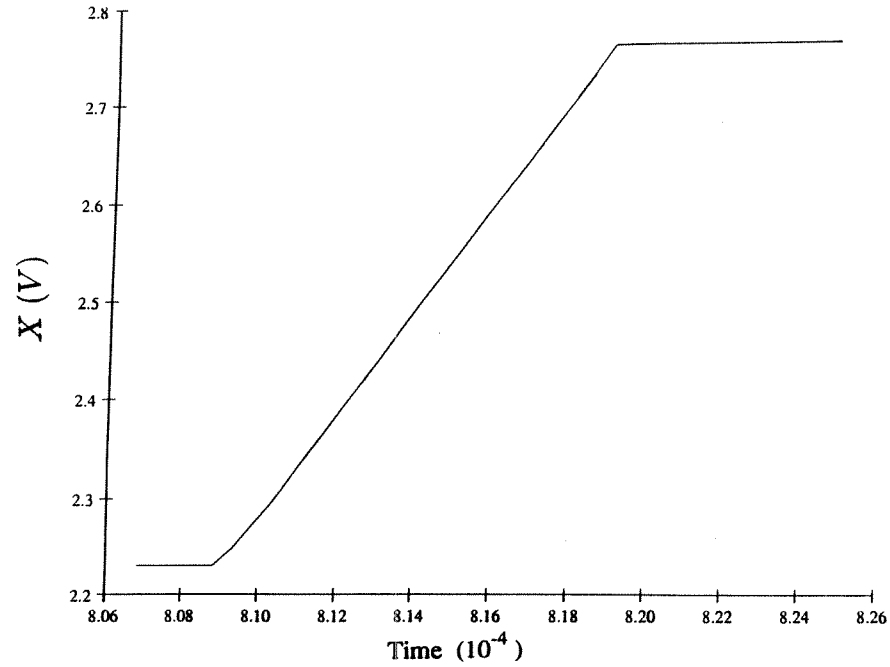

Figure 4.33: The convergence of the global optimization circuit to the global maxima as observed from the time evolution of state voltage X (see equation 4.59) for simulation S1.

Finally, when the state voltage X enters the region of $F(X) > F(X^*) + \epsilon$, the circuit enters the gradient ascent phase, in accordance with third line the of equations (4.57) and (4.59). The winner-take-all circuit blocks the repeller current by bringing it down to zero (Figure 4.30), and lets only the gradient current flow (Figure 4.31), which is to be integrated by the capacitor C as the output current I_{out} (Figure 4.32). Noting that the gradient ascent current I_{gr} is equal to $C\lambda\frac{dF[X]}{dX}$ (which also equals to $C\ dX/dt$), the spike seen in Figure 4.31 (and also in Figure 4.32) is due to the initially increasing and subsequently decreasing slope of the objective function at the left side of the bump that consists of the global maximum. The gradient ascent current I_{gr} asymptotically approaches the $0A$ level because of the decreasing gradient in the neighborhood of the global maximum (Figure 4.31). This enforces the output current to vanish also (Figure 4.32), such that the state voltage X at the terminals of the capacitor ceases to integrate any current. X thus converges to $2.77V$ with only $30mV$ error from the global maximum located $2.80V$, as seen in Figure 4.33. The error, which was expected, is minimal and is due to the low accuracy of local gradient ascent approximation on steep functions and is not related with the global ascent property displayed by the circuit.

Also note that the convergence is stable, which can also be seen from Figure 4.29 by the saturation of $F(X)$ in time due to the stable and constant state X.

Furthermore, the convergence time required for the circuit to start from the initial point and to converge to the global maximum is about 10 micro-seconds (Figure 4.33), which we believe to be remarkable.

Finally, we consider the drawback of the circuit due to the small region of operation in which both the repeller current I_{rep} and the gradient current I_{gr} flow simultaneously via the output current I_{out}. We already saw the origin of such operation in the second line of equations (4.56), (4.59), and discussed it in detail on pages 86, 91 and 96. Although we concluded from the simulations that this drawback does not affect the efficiency of the global optimization dynamics as long as the power of the repeller K is chosen sufficiently low, below we present a solution that completely eliminates the problem regardless of the

selection of K.

From Figures 4.23 and 4.24 we note that $I_{out} = I_{rep} + I_{gr}$ flows for the region which correspond to the condition determined by,

$$\begin{aligned} V_{1\_w} &> 5 - V_{th} = 1V \\ \text{and} \quad V_{2\_w} &> 5 - V_{th} = 1V \end{aligned} \tag{4.60}$$

where, V_{th} is the threshold voltage set to $4V$ in the modified winner-take-all circuit shown in Figures 4.22 and 4.25. It is clear that (see Figure 4.22) as $F(X)$ initially increases, when V_{1_w} reaches just above $1V$, transistor Q_a turns on, which causes premature flow of I_{gr} (i.e., when $F(X) < F(X^*)$) while I_{rep} is flowing. Similarly, as $F(X)$ further increases above $F(X^*)$, until V_{2_w} falls below $1V$, transistor Q_b is on, which permits I_{rep} to flow beyond the desired time (i.e., while $F(X) > F(X^*)$) in addition to the current I_{gr}. Hence we see that there exists an overlap between the currents I_{rep} and I_{gr}, whereas ideally a sharp transition as given in equations (4.35) and (4.54) was desired.

Such desired sharp transition could be achieved if the circuit allowed only I_{rep} to flow for $F(X)$ less than $F(X^*)$, and I_{gr} to flow otherwise. We found that this could be easily achieved by reducing the threshold voltage V_{th} of the inverters from the originally set value of $4V$ to the value given below,

$$V_{th} \simeq 5 - V_m \tag{4.61}$$

where V_m is as given in equation (4.44). It is important to note that (see Figure 4.22) V_m is the fixed voltage at the point where V_{1_w} and V_{2_w} intersect and are therefore equal. This point also corresponds to the condition of $F(X) = F(X^*)$. Setting V_{th} as in (4.61) will result in the following circumstances.

As $F(X)$ initially increases (i.e., when Q_2 is a winner during the repelling phase with Q_b on and $I_{out} = I_{rep}$), transistor Q_a will not turn on until the voltage V_{1_w} reached $5 - V_{th}$ due to the operation of the inverter that involves transistors Q_{in_a} and Q_{ip_a}. Furthermore, since V_{th} is set to the value given by equation (4.61), it follows that $5 - V_{th} = V_m$. Consequently, Q_a will not turn on until V_{1_w} reached V_m. Therefore, as long as $F(X) < F(X^*)$ only the

current I_{rep} flows and I_{gr} is blocked due to Q_a being off.

However, as soon as $F(X)$ increases just above $F(X^*)$ such that V_{1_w} reaches just above V_m, transistor Q_a turns on. At the same time V_{2_w} falls just below V_m (and hence $5 - V_{th}$) which forces Q_b to turn off instantaneously due to the inverter that involves the transistors Q_{in_b} and Q_{ip_b}. Hence the condition is reversed with I_{gr} flowing while I_{rep} is blocked.

In order to demonstrate this concept, we made another simulation using the modified winner-take-all circuit of Figure 4.22 in parallel to the one shown in Figures 4.23 and 4.24. The only parameter that needed to be modified was the fixed voltage V_{th}, which was dependent on the value of V_m as given in (4.61). V_m can be calculated to be $1.734V$ either by using equations (4.44) and (4.42) (with $V_{b\_w} = 0.7V$, $I_0 \simeq 0.068fA$ and $I_m = 70nA$) or via the simulation of Figure 4.23. This would require V_{th} to be set to $3.266V$, however, since the inverters were operating at above-threshold ranges (i.e., due to high gate-to-source voltages), reducing V_{th} even further and setting it to

$$V_{th} = 2.435V \tag{4.62}$$

gave the optimal results. The results of the simulation is shown in Figures 4.34 and 4.35. Comparing Figure 4.35 with Figure 4.24 demonstrates that the undesired region, where I_{rep} and I_{gr} overlaps, is completely eliminated and a sharp transition between repeller action and gradient ascent action is achieved as was originally desired. We conclude that when V_{th} is set according to (4.62), ϵ in the second line of equation (4.56) diminishes to zero, such that equation (4.56) (or (4.57)) converges to the desired form of equation (4.54), which is also repeated below

$$I_{out} = \begin{cases} I_{rep}, & \text{for } F(X) < F(X^*) \quad \text{i.e., repelling phase,} \\ I_{gr}, & \text{for } F(X) > F(X^*) \quad \text{i.e., maximization phase.} \end{cases} \tag{4.63}$$

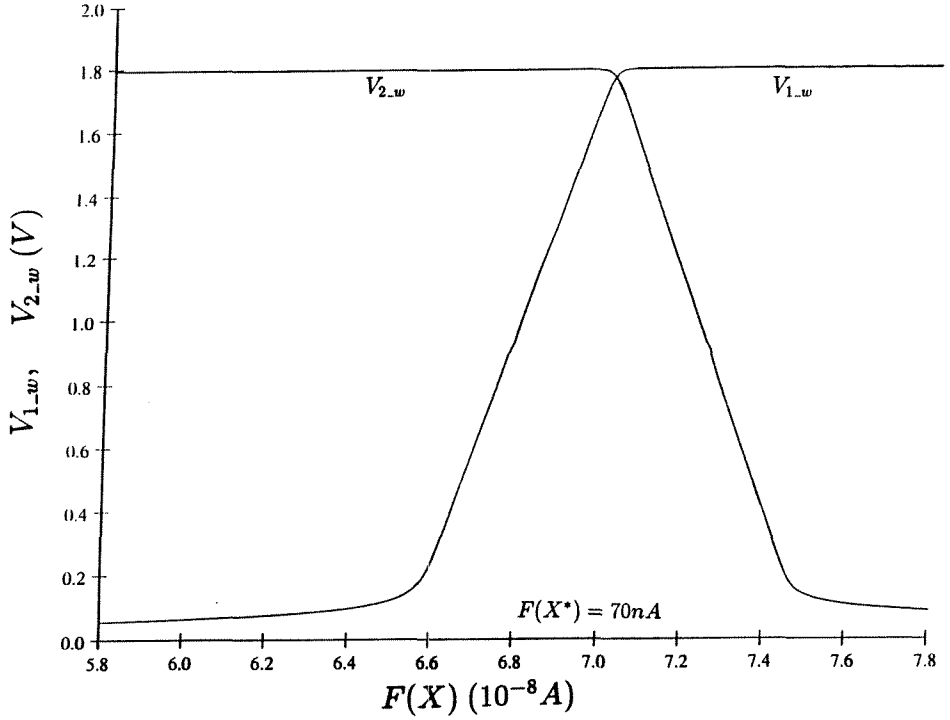

Figure 4.34: V_{1_w} vs. V_{2_w} characteristics for the modified winner-take-all circuit with $V_{th} = 2.435V$

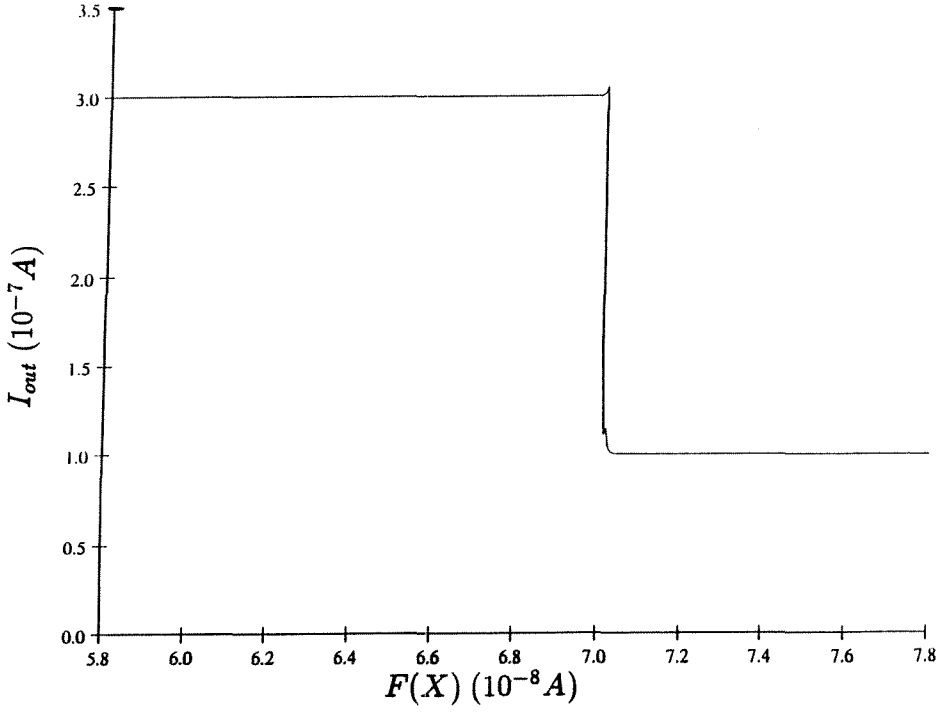

Figure 4.35: The output characteristics of the modified winner-take-all circuit for the simulation of Figure 4.34

Similarly, when global optimization is performed with the global circuit of Figure 4.25 using (4.62), the dynamics of the circuit does not undergo the limitation imposed by equation (4.59), but instead performs the ideal form of (4.35), given below,

$$\dot{X} = \begin{cases} K\,(X - X^*)^{1/3}, & F[X] - F[X^*] \leq 0 \quad \text{repelling phase} \\ \frac{dF[X]}{dX}, & F[X] - F[X^*] > 0 \quad \text{maximization phase.} \end{cases} \tag{4.64}$$

It is clear that this circuit with this new dynamics (which is obtained by a small modification of V_{th}) is not limited by the conditions of the previous dynamics, where a reasonably low value for the repeller power K had to be chosen in order to avoid instability (see discussion on page 96). The new dynamics can perform reliable global optimization for ANY value of the repeller power K due to the sharp transition achieved between I_{rep} and I_{gr} currents.

As a last remark, one should also note that since the global optimization simulation involved setting the winner-take-all bias voltage V_{b_w} to 1V instead of 0.7V; due to equation (4.44), V_{th} should be expected to be set to a lower value than the one given in (4.62) for optimal global ascent performance.

4.4 Summary

In this chapter we have presented Analog VLSI circuits that implement the concepts of terminal repeller and gradient descent. The performance of these circuits together with the measured data from the chips we have built, are discussed. These two circuits and a modified winner-take-all circuit have been employed to implement the TRUST algorithm and consequently a global optimization circuit has been designed. The simulated results demonstrate that hardware implementation of the TRUST formalism leads to dramatic speed enhancement in the context of global optimization.

Chapter 5

Conclusions

This thesis has introduced TRUST (Chapter 2 and [CBB93a]), a novel dynamical system with global descent property. As a result of this property, the TRUST formulation is shown to be very effective in globally optimizing energy or cost functions employed in global optimization problems.

The TRUST methodology formulates optimization as the solution of a single vector differential equation incorporating terminal repellers and a novel subenergy tunneling function. The flow of this dynamical system leads to global optimization. The algorithm is shown to be provably convergent to the global minimum in the one-dimensional case (Section 2.4).

Although convergence to a global minimum is not formally guaranteed in the multi-dimensional case (Section 2.5), the TRUST formalism systematically converged to the global minimum in all benchmark simulations (Section 2.6 and Appendix A), even in the multi-dimensional case.

Benchmark comparisons (Section 2.6) with other global optimization procedures have demonstrated that TRUST is significantly faster, as measured by the number of function evaluations, than the best currently available methods for these standard functions.

The number of function evaluations is only one criterion to be used in comparing this algorithm with other algorithms. It is important to emphasize that TRUST has a number of other advantages. First, while the algorithm is not guaranteed to find the global min-

na in multiple dimensions, it does have a global descent property. It is thus practically ıseful for multi-dimensional problems. For n-dimensional functions, the algorithm can be :omputed as the parallel solution of n weakly coupled differential equations. Consequently, he complexity and computational cost of the algorithm is not strongly dependent upon the ›roblem dimensionality. Second, this formulation naturally leads to a simple and compu- ationally efficient stopping criterion. Finally, TRUST is robust with respect to the basic ılgorithm parameters. Necessary conditions on the algorithm parameters were derived in Subsection 2.3.4.

The TRUST algorithm has already been employed, with encouraging results, to robotics applications [BCB91, CB92]. This thesis has introduced the adaptation of the TRUST formalism to artificial neural networks in order to eliminate the local minima problem during learning (Chapter 3 and [CBB93b]). Here, TRUST (or Global Descent) provides a simple extension to the Backpropagation algorithm by replacing the gradient descent method during training.

Testing Global Descent for common benchmark functions of the artificial neural network literature demonstrate that Backpropagation associated with Global Descent escapes encountered local minima, and in most cases converges to the globally minimal solution (Section 3.3).

Furthermore, the structure of the TRUST formulation makes it suitable for implementation in parallel Analog VLSI circuits. We have designed, fabricated and tested analog circuits for terminal repeller and gradient descent (Chapter 4), which are the main tools of the TRUST dynamical system.

Finally, by employing additional control logic to these circuits, a global optimization circuit (Section 4.3) is designed that implements the one-dimensional TRUST algorithm. This circuit, as demonstrated with the simulated experiments, can locate the global minimum of arbitrary one-dimensional functions with substantial speed enhancement.

Appendix A

Test Functions and Relevant Parameters used in Benchmark Studies

The functions used in the benchmark studies of Section 2.6 are listed below. For each function, we also summarize in tabular form the relevant parameters used in the benchmark simulations. In these tables, $\vec{x}_L$ and $\vec{x}_U$ are respectively lower and upper bounds of the domain of interest $\mathcal{D}$; $\vec{x}_I$ is the initial condition; $\vec{\epsilon}$ is the TRUST perturbation, Δt is the Euler integration step size, and k is the repeller power. In all simulations, TRUST used the same values for $\mathcal{D}$, $\vec{x}_I$ and Δt, as the methods to which its performance is compared.

Function 1. Two-Dimensional 6-Hump CambelBack Function

$$f(x_1, x_2) = [4 - 2.1x_1^2 + (x_1^4/3)]x_1^2 + x_1x_2 + (-4 + 4x_2^2)x_2^2.$$

Number of local minima : 6;

number of global minima: 2;

global minimum found by TRUST:

$$[x_{1GM}, x_{2GM}] = [0.08983, -0.71265], \quad \text{for } (i), (iv),$$
$$[x_{1GM}, x_{2GM}] = [-0.08983, 0.71265], \quad \text{for } (ii), (iii).$$

Table A.1: Benchmark parameters for Function 1.

Trial	x_{1L}	x_{1U}	x_{2L}	x_{2U}	x_{1I}	x_{2I}	ϵ_1	ϵ_2	Δt	k
(i)	-3	3	-2	2	-3.0	-2.0	0.01	0.01	0.01	10
(ii)	-3	3	-2	2	3.0	2.0	-0.01	-0.01	0.01	10
(iii)	-3	3	-2	2	-2.0	-1.0	0.01	0.01	0.10	10
(iv)	-3	3	-2	2	-1.6	0.9	0.01	-0.01	0.10	10

Function 2. Two-Dimensional Shubert Function

$$f(x_1, x_2) = \left\{\sum_{i=1}^{5} i\cos[(i+1)x_1 + i]\right\}\left\{\sum_{i=1}^{5} i\cos[(i+1)x_2 + i]\right\}$$

Table A.2: Benchmark parameters for Function 2.

Trial	x_{1L}	x_{1U}	x_{2L}	x_{2U}	x_{1I}	x_{2I}	ϵ_1	ϵ_2	Δt	k
(i)	-10	10	-10	10	-10	-10	0.1	0.01	0.0004	100
(ii)	-10	10	-10	10	-10	-10	0.1	0.01	0.0004	500
(iii)	-10	10	-10	10	1.0	1.0	-0.01	-0.1	0.0004	500

Number of local minima : 760;

number of global minima: 18;

global minimum found by TRUST:

$$[x_{1GM}, x_{2GM}] = [-7.08351, -7.70831], \quad \text{for } (i), (ii),$$
$$[x_{1GM}, x_{2GM}] = [-0.80032, -1.42513], \quad \text{for } (iii).$$

Function 3. N-Dimensional Test Function

$$f(\vec{x}) = \frac{1}{2}\sum_{j=1}^{N}\left(x_j^4 - 16x_j^2 + 5x_j\right),$$
$$\vec{x} = [x_1, x_2, \cdots, x_j, \cdots, x_N]$$

Number of local minima : 2^N;

number of global minima: 1;

global minimum found by TRUST:

$$[\vec{x}_{GM}] = [-2.90354, -2.90354, \cdots, -2.90354]$$

Table A.3: Benchmark parameters for Function 3.

Trial	N	x_{jL}	x_{jU}	$[\vec{x}_I]$	$[\vec{\epsilon}]$	Δt	k
(i)	1	-5	5	[5]	$[-0.01]$	0.02	10
(ii)	2	-5	5	[5, 5]	$[-0.01, -0.01]$	0.02	10
(iii)	4	-5	5	[5, .., 5]	$[-0.01, .., -0.01]$	0.02	10
(iv)	10	-5	5	[5, ..., 5]	$[-0.01, ..., -0.01]$	0.02	10

Function 4. Two-Dimensional Test Function

$$f(x_1, x_2) = 0.5x_1^2 + 0.5\,[1 - \cos(2x_1)] + x_2^2,$$

Table A.4: Benchmark parameters for Function 4.

Trial	x_{1L}	x_{1U}	x_{2L}	x_{2U}	x_{1I}	x_{2I}	ϵ_1	ϵ_2	Δt	k
(i)	-3	3	-3	3	-3	0	0.01	0.01	0.1	10
(ii)	-3	3	-3	3	-3	-3	0.01	0.01	0.1	10

Number of local minima : several;

number of global minima: 1;

global minimum found by TRUST:

$$[x_{1GM}, x_{2GM}] = [0, 0].$$

Function 5. Two-Dimensional Test Function

$$f(x_1, x_2) = 10^n x_1^2 + x_2^2 - (x_1^2 + x_2^2)^2 + 10^m (x_1^2 + x_2^2)^4, \quad n = -m.$$

Number of local minima : > 3;

number of global minima: 2;

global minimum found by TRUST:

$$[\vec{x}_{GM}] = [0, 1.38695], \quad \text{for } n = 1,$$
$$[\vec{x}_{GM}] = [0, 2.60891], \quad \text{for } n = 2,$$
$$[\vec{x}_{GM}] = [0, 4.70174], \quad \text{for } n = 3,$$
$$[\vec{x}_{GM}] = [0, 8.39401], \quad \text{for } n = 4.$$

Table A.5: Benchmark parameters for Function 5.

Trial	n	x_1	x_{2L}	x_{2U}	x_{1I}	x_{2I}	ϵ_1	ϵ_2	Δt	k
(i)	1	0	-2	2	0	1	0.01	0.01	0.1	10
(ii)	2	0	-4	4	0	1	0.01	0.01	0.01	10
(iii)	3	0	-5	5	0	1	0.01	0.01	0.004	10
(iv)	4	0	-10	10	0	1	0.01	0.01	0.0006	10

Function 6. The Two-Dimensional Rastrigin Function

$$f(x_1, x_2) = x_1^2 + x_2^2 - \cos(18x_1) - \cos(18x_2).$$

Number of local minima : 50;

number of global minima: 1;

global minimum found by TRUST:

$$[x_{1GM}, x_{2GM}] = [0, 0].$$

Table A.6: Benchmark parameters for Function 6.

Trial	x_{1L}	x_{1U}	x_{2L}	x_{2U}	x_{1I}	x_{2I}	ϵ_1	ϵ_2	Δt	k
	-1	1	-1	1	-1	-1	0.01	0.01	0.01	10

Function 7. Two-Dimensional Branin Function

$$f(x_1, x_2) = \left(x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6\right)^2 + 10(1 - \frac{1}{8\pi})\cos x_1 + 10.$$

Number of local minima : 3;

number of global minima: 3;

global minimum found by TRUST:

$$[x_{1GM}, x_{2GM}] = [3.14158, 2.27505].$$

Table A.7: Benchmark parameters for Function 7.

Trial	x_{1L}	x_{1U}	x_{2L}	x_{2U}	x_{1I}	x_{2I}	ϵ_1	ϵ_2	Δt	k
	-5	10	0	15	-5	0	0.01	0.01	0.1	10

Function 8. One-Dimensional Test Function

$$f(x) = \sin x + \sin(10x/3)x + \log x - 0.84x$$

Number of local minima : 3;

number of global minima: 1;

global minimum found by TRUST:

$$x_{GM} = 5.19978.$$

Table A.8: Benchmark parameters for Function 8.

Trial	x_L	x_U	x_I	ϵ_x	Δt	k
	2.7	7.5	2.7	0.1	0.16	10

Function 9: One-Dimensional Test Function

$$f(x) = -\left\{\sum_{i=1}^{5} \sin[(i+1)x + i]\right\}.$$

Number of local minima : 20;

number of global minima: 3;

global minimum found by TRUST:

$$x_{GM} = -6.72004, \quad \text{for } (i),$$

$$x_{GM} = 5.84633, \quad \text{for } (ii).$$

Table A.9: Benchmark parameters for Function 9.

Trial	x_L	x_U	x_I	ϵ_x	Δt	k
(i)	-10	10	-10	0.1	0.06	10
(ii)	-10	10	10	-0.1	0.04	10

References

[AK] B. P. Anderson and D. Kerns. Using noise injection and correlation in analog hardware to esimate gradients. Submitted to IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications.

[And72] R. S. Anderssen. Global optimization. In R. S. Anderssen, L. S. Jennings, and D. M. Ryan, editors, *Optimization*, page 26. University of Queensland Press, St. Lucia, 1972.

[APPZ85] F. Aluffi-Pentini, V. Parisi, and F. Zirilli. Global optimization and stochastic differential equations. *Journal of Optimization Theory and Applications*, 47:1–15, 1985.

[Ati91] A. Atiya. *Learning Algorithms for Neural Networks.* Ph.D. thesis, Caltech, Pasadena, CA, 1991.

[BCB91] J.W. Burdick, B.C. Cetin, and J. Barhen. Efficient global redundant configuration resolution via subenergy tunneling and terminal repelling. In *Proc. IEEE Inter. Conf. on Robotics and Automation*, Sacramento, CA, April 1991.

[Blu89] E.K. Blum. Approximation of Boolean functions by sigmoidal networks: Part i: Xor and other two-variable functions. *Neural Computation*, 1:532–540, 1989.

[Bre70] H. A. Bremermann. Method of unconstrained global optimization. *Mathematical Biosciences*, 9:1–15, 1970.

[Bro58] S. H. Brooks. A discussion of random methods for seeking maxima. *Operations Research*, 6:244, 1958.

[BTG90] J. Barhen, N. Toomarian, and S. Gulati. Application of adjoint operators to neural learning. *Applied Math Letters*, 3:13–18, 1990.

[BZT90a] J. Barhen, M. Zak, and N. Toomarian. Adjoint operator algorithms for faster learning in neural networks. *Advanced Neural Information Processing Systems*, 2:498–508, 1990.

[BZT90b] J. Barhen, M. Zak, and N. Toomarian. Non-Lipschitzian neural dynamics. In R. Eckmiller, editor, *Advanced Neural Computers*, pages 102–112. North-Holland, 1990.

[CB92] I. Chen and J.W. Burdick. Finding antipodal point grasps on irregularly shaped objects. In *Proc. IEEE Inter. Conf. on Robotics and Automation*, Nice, France, May 1992.

[CBB93a] B.C. Cetin, J. Barhen, and J.W. Burdick. Terminal repeller unconstrained subenergy tunneling (trust) for fast global optimization. *Journal of Optimization Theory and Applications*, 77(1), April 1993.

[CBB93b] B.C. Cetin, J.W. Burdick, and J. Barhen. Global descent replaces gradient descent to avoid local minima problem in learning with artificial neural networks. In *1993 IEEE International Conference on Neural Networks*, San Francisco, CA, March 1993.

[Day90] J. Dayhoff. The exclusive-or: A classic problem. In *Neural Network Architectures: an Introduction*, pages 76–79. VNR Press, New York, NY, 1990.

[Ge90] R. Ge. A filled function method for finding a global minimizer of a function of several variables. *Mathematical Programming*, 46:191–204, 1990.

[GS90] W. A. Gruver and E. Sachs. *Algorithmic Methods in Optimal Control.* Pitman Publishing Ltd., Melbourne, 1990.

[HKP91] J. Hertz, A. Krogh, and R.G. Palmer. Back-propagation: Examples and applications. In *Introduction to the Theory of Neural Computation*, pages 130–141. Addison-Wesley, 1991.

[HS86] G.E. Hinton and T.J. Sejnowski. Learning and relearning in Boltzmann machines. In D.E. Rumelhart, J.L. McClelland, and the PDP research group, editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, chapter 7, pages 282–317. MIT Press, Cambridge, MA, 1986.

[KGV83] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.

[Kir93] D. B. Kirk. *Accurate and Precise Computation using Analog VLSI, with Applications to Computer Graphics and Neural Networks.* Ph.D. thesis, California Institute of Technology, 1993.

[KT85] A. H. Kan and G. T. Timmer. A stochastic approach to global optimization. In P.T. Boggs and R.B. Byrd, R.H.and Schnabel, editors, *Numerical Optimization*, pages 245–262. SIAM, Philadelphia, Pennsylvania, 1985.

[LM85] A. V. Levy and A. Montalvo. The tunneling algorithm for the global minimization of functions. *SIAM Journal on Scientific and Statistical Computing*, 6:15–29, 1985.

[Mea89] C. Mead. *Analog VLSI and Neural Systems*, pages 98–99. Addison Wesley, Menlo Park, CA, 1989.

[MHB89] J.M. McInerney, K.G. Haines, and S. Biafore. Error surfaces of multi-layer networks can have local minima. Technical Report CS89–157, UCSD, CA, 1989.

[MP69] M. Minsky and S. Papert. *Perceptrons.* MIT press, Cambridge, MA, 1969.

[MRTT53] N. Metropolis, A. W. Rosenbluth, A. H. Teller, and E. Teller. Equation of state calculation by fast computing machines. *Journal of Chemical Physics*, 21:1087, 1953.

[Pri78] W. L. Price. A controlled random search procedure for global optimization. In L. C. W. Dixon and G.P. Szego, editors, *Towards Global Optimization 2.* North-Holland, Amsterdam, 1978.

[RHW86] D.E. Rumelhart, G.E. Hinton, and R.J. Williams. Learning internal representations by error propagation. In D.E. Rumelhart, J.L. McClelland, and the PDP research group, editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, chapter 8, pages 318–362. MIT Press, Cambridge, MA, 1986.

[SD88] J. Sietsma and R. J. F. Dow. Neural net pruning—why and how. In *IEEE International Conference on Neural Networks Vol. I*, pages 325–333, San Diego 1988, 1988. New York: IEEE.

[SH87] H. Szu and R. Hartley. Fast simulated annealing. *Physics Letters A*, 122:157–162, 1987.

[Tor78] A. A. Torn. A search clustering approach to global optimization. In L. C. W. Dixon and G.P. Szego, editors, *Towards Global Optimization 2.* North-Holland, Amsterdam, 1978.

[TZ89] A. Torn and A. Zilinskas. *Global Optimization.* Springer-Verlag, Berlin, 1989.

[UD89] C.B. Umminger and S.B. DeWeerth. Implementing gradient following in analog vlsi. In *Decennial CIT Conference on VLSI*, 1989.

[vLPL+88] A. von Lehman, E. G. Paek, P. F. Liao, A. Marrakchi, and J. S. Patel. Factors influencing learning by back-propagation. In *IEEE International Conference on Neural Networks Vol. I*, pages 335–341, San Diego 1988, 1988. New York: IEEE.

[Was89] P.D. Wasserman. Backpropagation and cauchy training: an overview. In *Neural Computing: Theory and Practice*, pages 87–92. VNR Press, New York, NY, 1989.

[WHS85] G. W. Walster, E. R. Hansen, and S. Sengupta. Test results for a global optimization problem. In P.T. Boggs, R.H. Byrd, and R.B. Schnabel, editors, *Numerical Optimization*, pages 272–287. SIAM, Philadelphia, Pennsylvania, 1985.

[Yao89] Y. Yao. Dynamic tunneling algorithm for global optimization. *IEEE Transactions on Systems, Man, and Cybernetics*, 19:1222–1230, 1989.

[Zak89] M. Zak. Terminal attractors in neural networks. *Neural Networks*, 2:258–274, 1989.

[ZB90] M. Zak and J. Barhen. Neural networks with creative dynamics. *Math. & Comp. Modeling*, 14:290–294, 1990.