ANALOG SOLUTION OF NP-HARD PROBLEMS

Michael G. Main

CU-CS-700-94 January, 1994

University of Colorado at Boulder
DEPARTMENT OF COMPUTER SCIENCE

ANALOG SOLUTION OF NP-HARD PROBLEMS

CU-CS-700-94 January, 1994

Michael G. Main

Department of Computer Science
University of Colorado at Boulder
Campus Box 430
Boulder, Colorado 80309-0430 USA

Research Notes

Analog Solution of NP-Hard Problems

Michael G. Main
(main@cs.colorado.edu)
Computer Science Department
University of Colorado at Boulder
Jan 5, 1994

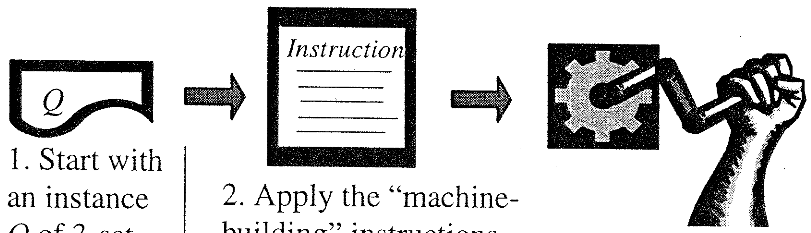
Abstract

This report shows a new NP-hardness result: Let *f* be a continuous piecewise-quadratic function with continuous partial first derivatives, such that all partial first derivatives are zero at the origin. The problem of determining whether the origin is a local minimum for *f* is NP-hard. The report also proposes an electronic analog machine to solve this NP-hard problem, and suggests some experiments to determine the efficiency of the machine.

Some update notes from June 1997 appear on page 23. Please send email if you would like a fuller report later in 1997.

Analog machines which solve NP-hard problems have been proposed but, as far as I know, an actual machine has never been built. In the example that I am most familiar with [1], the proposed machine is intriguing: The authors start with any instance of 3-satisfyability, and from the specification of the instance they describe how to build a contraption of gears and other mechanical parts. One of the gears in the machine has a crank handle attached to it. The analysis in [1] shows that the handle can be turned *iff* the given instance of 3-sat is satisfiable.

So, in order to solve an instance Q of 3-satisfyability, one follows these steps:



1. Start with an instance Q of 3-sat.

2. Apply the "machine-building" instructions to Q, resulting in a "machine for Q."

3. The machine for Q has a crank. If the crank can turn, then Q is satisfiable; otherwise Q is not satisfiable.

The instructions for building the "machine for Q" make it clear that the resulting machine has only a polynomial number of parts, and a polynomial number of connections between the parts. (*I.e.,* The number of parts and connections in the machine is bounded by a polynomial function $f(n)$, where n is the number of clauses in the instance Q.) However a number of other questions are left unanswered: *Is it really tractable to build and operate such a machine for interesting instances? Or will practical considerations foil the plan, such as an unattainable accuracy in assembling the parts, or an exponential force required to actually turn the handle? What occurs in a digital simulation of the analog machine?*

This report is a first step toward answering these questions — not directly for the mechanical crank machine, but for an electronic version of the crank machine which can be built from off-the-shelf components. The outline of the paper is as follows:

Section 1 provides some standard background material on how electronic analog machines solve dynamical systems.

Section 2 shows how to build a dynamical system corresponding to any instance of a particular NP-hard problem. A proposal is made for an electronic analog machine which solves this dynamical system, and in the process answers the instance of the corresponding NP-hard problem— although the proof that the instance has been correctly answered requires us to accept the *Downhill Principle* from [1]. The machine's size is polynomial (in the size of the instance of the NP-hard problem), but it is uncertain whether its running time (or other resources) will be polynomial, and it is also uncertain whether the Downhill Principle is valid in practice.

Sections 3 discuss a programmable version of the machine from Section 2. The programmable version can be digitally programmed (by setting relays) to solve any instance of a variation of 3-sat, up to some fixed size n.

1. Dynamical Systems and Analog Machines

Consider a surface in n-dimensional real space, defined by a continuous differentiable function $f: \Re^n \rightarrow \Re$. The value of the function f at a point $\mathbf{x} \in \Re^n$, gives the "height" of the surface at point $\mathbf{x}$. If we place a ball on the surface with a known initial position and velocity, and apply a constant downward force, then the ball will roll. When the ball is at a location $\mathbf{x}$, the instantaneous acceleration in n-dimensional space is given by the vector:

$$-\nabla f(\mathbf{x}) \frac{F}{m}$$

The function ∇f is the gradient of the surface defined by f; the mass of the ball is m; and the magnitude of the downward force is F. Normally we will choose F/m to be one so that the acceleration at a point $\mathbf{x}$ is simply $-\nabla f(\mathbf{x})$. In terms of derivatives of f, the magnitude of the acceleration in a fixed direction x is $-\partial f/\partial x$.

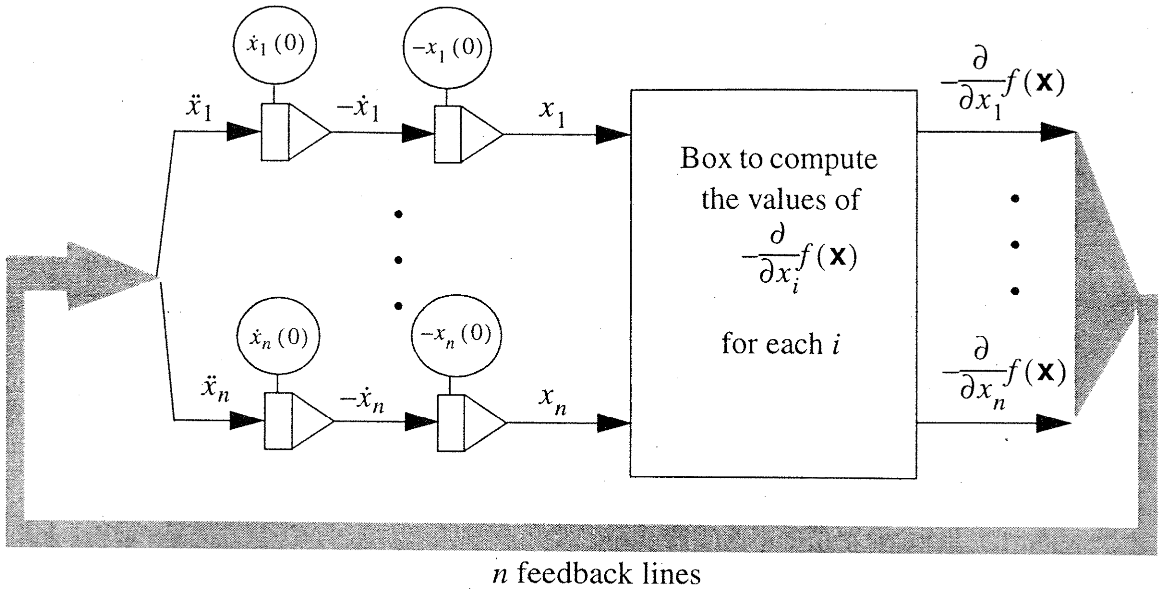
The function f, together with the initial position and velocity of the ball, is an example of a *dynamical system*. For these dynamical systems, we use the notation $x_i(t)$ to denote the position of the ball at time t along the i^{th} axis ($i \in \{1, \ldots, n\}$). Similarly, $\dot{x}_i(t)$ is the speed, and $\ddot{x}_i(t)$ is the acceleration (at time t along the i^{th} axis). Following the usual convention, we will omit the argument (t) when there is no possibility of confusion. In general, we would like to be able to "solve the system" — in other words, determine the values of x_i, $\dot{x}_i$ and $\ddot{x}_i$ for each i and at any time t.

Often, the system can be solved analytically by solving the system of n simultaneous partial differential equations (using $\mathbf{x}$ for the vector $\langle x_1, \ldots, x_n \rangle$):

$$\left.\begin{aligned} \ddot{x}_i &= -\frac{\partial}{\partial x_i} f(\mathbf{x}) \\ \dot{x}_i(0) &= \text{initial velocity in direction } i \\ x_i(0) &= \text{initial position along direction } i \end{aligned}\right\} i \in \{1, \ldots, n\}$$

Lacking an analytical solution, we can sometimes build an analog machine to solve the system of pde's. In effect, the analog machine simulates the dynamical system of a ball rolling on an n-dimensional surface, under the influence of a constant downward force. This machine is not new, but is standard in textbooks such as [4]. Among other things, the analog machine has three wires for each of the n dimensions. As the machine runs, the voltages on the $3n$ wires represent the ball's position, velocity and acceleration for each of the n dimensions. All the values are maintained in real time, starting at some initial time t=0.

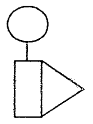
The particular construction we have in mind works for any n-dimensional surface where the height of the surface is given by a function $f: \Re^n \rightarrow \Re$ such that all the partial



$\dot{x}_1(0)$
$-x_1(0)$
$\ddot{x}_1$
$-\dot{x}_1$
x_1
Box to compute the values of $-\frac{\partial}{\partial x_i}f(\mathbf{x})$ for each i
$-\frac{\partial}{\partial x_1}f(\mathbf{x})$
$\dot{x}_n(0)$
$-x_n(0)$
$\ddot{x}_n$
$-\dot{x}_n$
x_n
$-\frac{\partial}{\partial x_n}f(\mathbf{x})$
n feedback lines

Figure 1. Machine to track position of a ball rolling on the surface defined by f

first derivatives of f exist and are computable by an electronic analog device. Figure 1 shows the design of the machine, using notation from [4].

Each of the symbols  is an electronic integrator. The initial output (on the right side of the symbol) is -1 volts times the initialization value (which appears inside the circle). After initialization, the output is maintained as -1 times the integral (over time) of the input (with the constant of integration determined by the initialization constant).

Our plan is to start this machine at some time t=0. As the machine runs, the voltages on the x_i , $-\dot{x}_i$ and $\ddot{x}_i$ lines keep track of the position, negative velocity and acceleration of the simulated ball rolling on the surface defined by f (with a specified initial position and velocity). The machine works correctly because the integral of the acceleration is the velocity, and the integral of the velocity is the position.

Example 1.

As a simple example, consider the function $f{:}\,\Re \rightarrow \Re$ defined by $f(x) = x - (x^2/2)$. This defines a one-dimensional surface drawn in Figure 2. A ball on the surface with a horizontal coordinate of x and a downward force applied will undergo a horizontal acceleration of $-f'(x)\frac{F}{m}$, where $f'(x) = 1 - x$ is the first derivative of the

$$f(x) = x - \frac{x^2}{2}$$

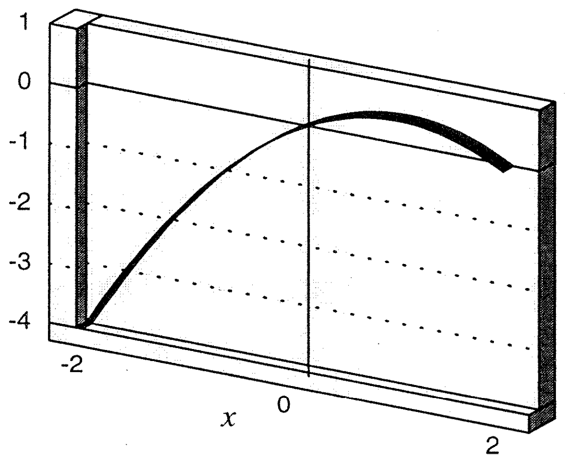


Figure 2. The one-dimensional surface defined by $f(x) = x - (x^2/2)$

surface, F is the magnitude of the downward force, and m is the mass of the ball. Therefore, a ball placed at the origin undergoes an initial acceleration of $-F/m$, and begins moving left on the surface.

To make this example concrete, suppose that F/m is unity, so that the initial acceleration is -1. The initial position is 0, as is the initial velocity, giving the following differential equation with initial values:

$$\ddot{x}(t) = x(t) - 1$$
$$x(0) = 0, \quad \dot{x}(0) = 0$$

In these equations, $x(t)$ is the position of the ball along the x-axis at time t, whereas $\dot{x}(t)$ and $\ddot{x}(t)$ are the horizontal speed and acceleration at time t. This differential equation can be solved analytically, with the solution:

$$x(t) = -\frac{e^t}{2} - \frac{e^{-t}}{2} + 1$$

$$\dot{x}(t) = -\frac{e^t}{2} + \frac{e^{-t}}{2}$$

$$\ddot{x}(t) = -\frac{e^t}{2} - \frac{e^{-t}}{2} = x(t) - 1$$

These equations gives the horizontal position, velocity and acceleration of the ball as a function of time.

The machine to simulate the movement of the ball on this surface is shown in Figure 3. The two triangles on the right side of the diagram are analog adders which have an output of -1 times the sum of all inputs. The open circle labeled -1 is a fixed input of minus 1 units (*e.g.*, -1 volts if we are using volts as our unit of measurement). The feedback line on the right is maintained at $-f'(x) = x - 1$. To run the machine, the voltages on the outputs of the two integrators is dropped to zero at time t=0. As the machine runs, the x coordinate of the simulated ball is obtained from the voltage on the line marked x. In this example, when the machine starts, the feedback line will quickly drop to -1, which pulls down the $\ddot{x}$ line. In turn, this starts to pull up the $-\dot{x}$ line, which pulls down the x line.

The actual hardware to build this analog machine is shown in Figure 4, using designs from pages 103 and 106 of [4]. Each analog integrator is built from an operational amplifier, a resistor and a capacitor, plus circuitry designed to clamp the output voltage to zero at initialization. The adders are constructed from resistors and an op amp. A digital simulation indicates that after 4 seconds, the voltage on line x will be around -26 volts, although in practice the circuit will saturate before then, leveling off at some negative value determined by the voltage supplied to the op amps. The value of x at t=4 in the analytical solution is $\left(-\frac{e^{4}}{2}-\frac{e^{-4}}{2}+1\right) \cong 26.3$.

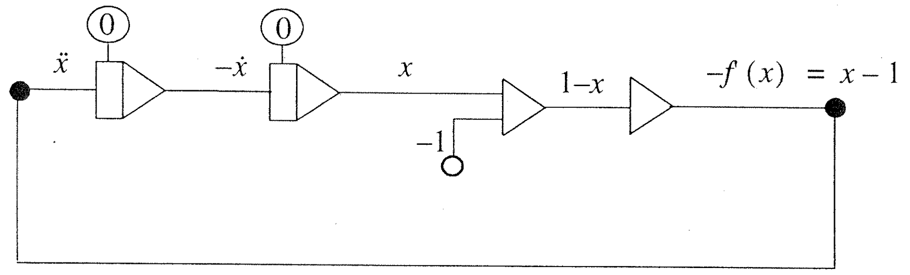
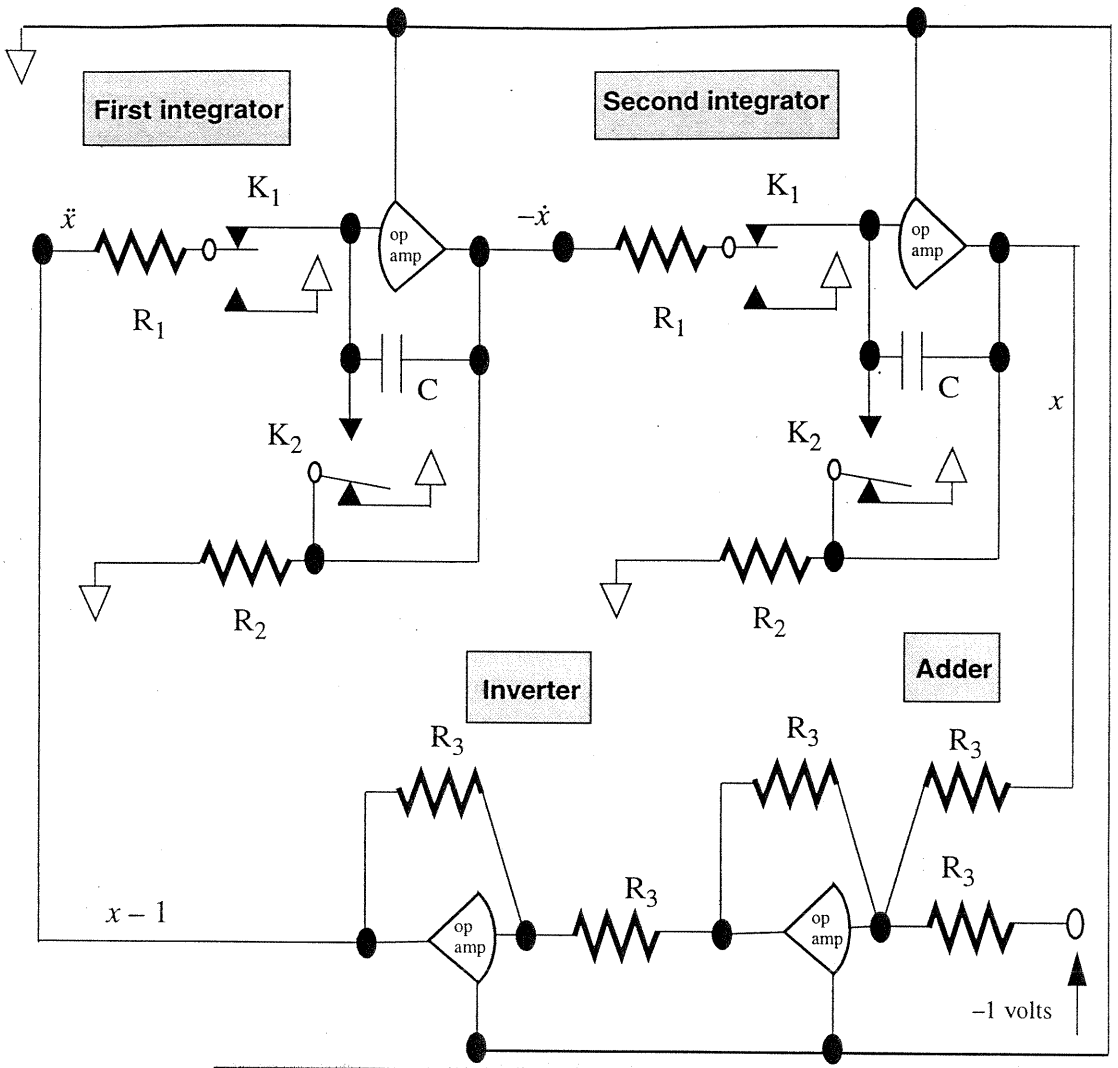


Figure 3. Analog machine to simulate movement of a ball on $f(x) = x - (x^2/2)$



- The relays K_1 (energized) and K_2 (de-energized) are set in the "run mode". Change relays to other positions for the "initialize mode".

- I used National Semiconductor LM741 op amps.

- The capacitances are all 10^{-5} farads, and the resistances are 10^5 ohms.

Figure 4. Implementation of the Analog machine from Figure 3

Example 2: An Unstable Equilibrium

As a second example, consider the one-dimensional surface defined by the equation $f(x) = -x^2/2$, shown in Figure 5. As before, place a ball on the surface at a location x and apply a downward force. The acceleration along the x-axis will be $-\frac{F}{m}f'(x)$ (where F is the downward force and m is the mass of the ball). Once again, take F/m to be unity, so that the equations describing the motion of the ball are:

$$\ddot{x}(t) = -f'(x) = x(t)$$

$$x(0) = \text{initial position}, \quad \dot{x}(0) = \text{initial velocity}$$

If the initial position and velocity are both zero, then the analytical solution shows that the position, velocity and acceleration stay precisely at zero for all t. This indicates that the origin of the surface is an equilibrium point — in other words, the gradient (or first derivative) is zero, and in a perfect world the ball stays put.

On the other hand, consider the analytical machine to solve this dynamical system, and provide initial values of zero. The machine is the same as that shown in Figures 3 and 4, but with the input of –1 volts changed to zero. The actual system will not "stay put", because noise in the system will be enough to kick it off the equilibrium point in a "downhill" direction. We don't know whether the noise will send the system left or

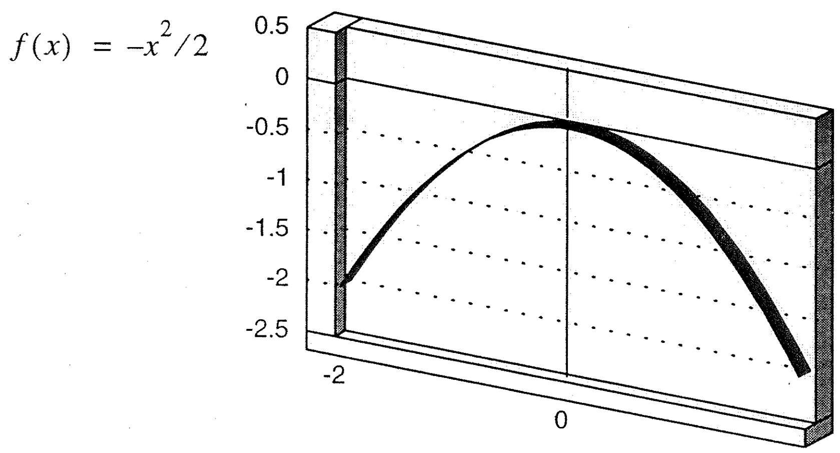


Figure 5. The one-dimensional surface defined by $f(x) = -x^2/2$

right, but there will be acceleration. This is because the equilibrium point is unstable. A small perturbation results in acceleration away from the equilibrium point.

For example, suppose that the actual initial x value is some small positive value $x(0)=\varepsilon$. Then, according to the analytical solution, we will have the following position, velocity and acceleration, as a function of time:

$$x(t) = \ddot{x}(t) = \frac{\varepsilon}{2}e^{t} + \frac{\varepsilon}{2}e^{-t} \qquad \dot{x}(t) = \frac{\varepsilon}{2}e^{t} - \frac{\varepsilon}{2}e^{-t}$$

So, even a small amount of noise results in a position which moves exponentially far from the origin as a function of time. This is good news, because if we build the system, then we can quickly tell that the point is unstable. *It's important to study what kind of unstable equilibria have this kind of "quick" unstable behavior in the presence of a small amount of noise.*

A principle related to this question is the "Downhill Principle" in Section 5 of the paper by Vergis, *et. al.* [1]. In effect, the principle states that if there is a downhill direction leading away from an equilibrium point, then an analog machine will eventually fall down such a downhill path. But there is no explicit analysis of the amount of time or energy required to "eventually" fall downhill.

Example 3: A Stable Equilibrium

In contrast to the last example, the surface defined by $f(x) = x^2/2$ has a stable equilibrium at the origin, as shown in Figure 6. In the absence of any systematic disturbance, a ball placed near the origin will stay near the origin. For example, the analytical solution shows that a small positive initial position of $x(0)=\varepsilon$ results in a periodic back and forth motion near the origin:

$$x(t) = \varepsilon\cos t \qquad \dot{x}(t) = -\varepsilon\sin t \qquad \ddot{x}(t) = -\varepsilon\cos t$$

The analog machine that solves this system is the same as that in Figures 3 and 4, but with the inverter omitted and the -1 input voltage changed to zero.

Of course, if there is a large amount of noise, or even small systematic disturbances, then the position might stray from the origin. This question of "too much noise" will be further examined in a moment.

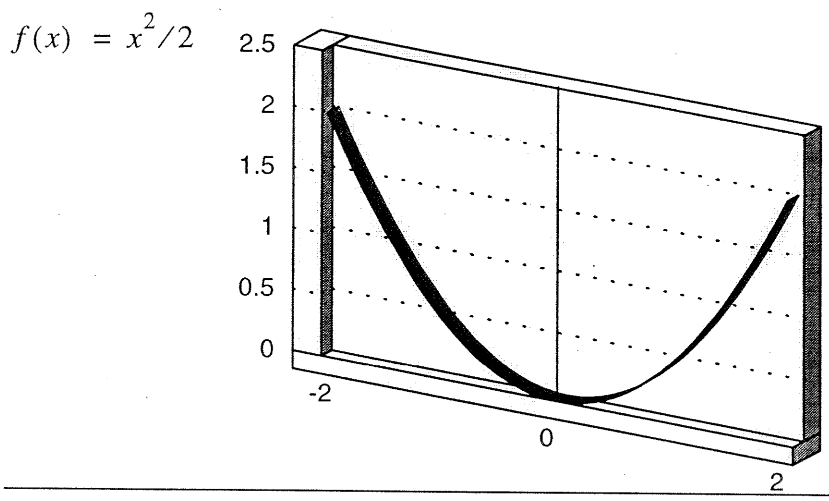


Figure 6. The one-dimensional surface defined by $f(x) = x^2/2$

Equilibria in Higher Dimensions

Thus far, the example dynamical systems have been one-dimensional functions defined by a function $f: \Re \rightarrow \Re$. But the same ideas apply to a higher dimensional surface defined by a function $f: \Re^n \rightarrow \Re$. Under "appropriate circumstances," an analog machine to solve a dynamical system with an equilibrium point should (1) Stay near a stable equilibrium point, and (2) Fall away from an unstable equilibrium point. An intuitive understanding of "appropriate circumstances" is that there is not enough noise to kick the machine away from a stable equilibrium, but there is sufficient noise to rapidly find a downhill gradient leading away from an unstable equilibrium.

The next section shows how to build electronic machines which solve instances of an NP-complete problem by determining whether an equilibrium point for a particular function is stable. The section also discusses the practicality of such a machine — that is, whether the "appropriate circumstances" are present to permit the machine to tell the difference between a stable and unstable equilibrium.

2. Analog Machines for an NP-Complete Problem

Consider the NP-complete problem Exact 1-in-3-Sat, defined here:

Let V be a set of Boolean variables. A *3-clause* is any set of three of these Boolean variables. An *assignment* for V is a function $A{:}V \to \{true, false\}$. For an assignment A and a variable u, we say that *u is a true variable* if $A(u) = true$; otherwise u is a false variable.

Exact 1-in-3-Sat: Given a finite set C of 3-clauses, does there exist an assignment such that each of the 3-clauses contains *exactly* one true variable?

This problem was shown to be NP-complete in [3]. Appendix A of this report shows how to take an instance P of Exact 1-in-3-Sat and construct a function $f_P{:}\Re^n \to \Re$, where the dimension n is one more than the number of variables in P. The function f_P has an equilibrium point at the origin. Moreover:

$$P \text{ is satisfyable} \Leftrightarrow \text{the equilibrium point is unstable}$$

The function f_P is a continuous piecewise-quadratic function, with continuous partial first derivatives. Since f_P is piecewise-quadratic, each partial first derivative $\partial f/\partial x$ is a piecewise linear function:

$$\frac{\partial f_P}{\partial x}{:}\Re^n \to \Re \qquad \text{is piecewise linear}$$

This means that the analog machine in Figure 1 can be easily built for the function f_P. The "ease" comes from the fact that the "black box" labeled "Box to compute the values of $-\frac{\partial}{\partial x_i} f(\mathbf{x})$ for each i" will be computing piecewise linear functions, and such functions are easy for an electronic analog machine (easier and more precise than quadratics or other non-linear functions).

Let's consider the behavior of this analog machine when it is started at the equilibrium point (*i.e.,* all positions and velocities are zero). We will call the machine M_P, since its construction depends on the instance *P*.

If the Instance *P* is Not Satisfiable

If the instance *P* is not satisfiable, then the origin is a local minimum for f_P, which is a stable equilibrium. Therefore, if we start M_P at the equilibrium point, then all values should stay near the equilibrium. It is possible that sufficient noise would kick the machine away from the equilibrium point, although the following property indicates that the origin is in fact a global minimum:

THEOREM 1: Let *P* be an unsatisfyable instance of 1-IN-3-SAT, and let f_P be the polynomial defined in Appendix A. Then for any vector The value of f_P is everywhere non-negative.

PROOF: Let $\mathbf{x}$ be a vector such that $f_P(\mathbf{x}) < 0$. Among other things, the proof of Theorem 1 showed how to construct a valid assignment for *P* from the vector $\mathbf{x}$.

❑

As a consequence of this result, when *P* is unsatisfyable and we start the analog machine at the equilibrium point, and watch the values of $\mathbf{x}$ (on the appropriate lines), we can compute $f_P(\mathbf{x})$ as the machine runs, and this value will never be negative.

If the Instance *P* is Satisfiable

If the instance *P* is satisfiable, then the origin is not a local minimum for f_P, and therefore the origin is an unstable equilibrium. In other words, there are points arbitrarily close to the origin where f_P has negative values (is "downhill"). In fact, an examination f_P shows that there is at least one direction leading away from the origin with a value that falls off proportional to the square of the distance from the origin:

THEOREM 2: Let P be a satisfiable instance of 1-IN-3-SAT, and let f_P be the polynomial defined in the proof of Appendix A. Let $\mathbf{x}$ be a vector determined from a valid assignment to P as follows: (1) The w component of $\mathbf{x}$ is 1, and (2) for each variable v_i in the instance P, the v_i component of $\mathbf{x}$ is 2 if v_i is true in the valid assignment, otherwise v_i is -2. Then for any scalar s:

$$f_P(s\mathbf{x}) = \frac{-s^2}{5}$$

PROOF: Each of the (a)- and (b)-terms of $f_P(s\mathbf{x})$ is zero, leaving only the (c)-term which is $-w^2/5$ (*i.e.*, $-s^2/5$).

❑

The consequence of this result is not entirely clear. Certainly, once the downhill path is found, the value of f_P will drop exponentially fast (as in Example 2 of the previous section). But how long will it take for noise to kick it onto a downhill path? Each downhill path which corresponds to a valid assignment in P is surrounded by a small volume where the negative gradient is quadratically steep (as in Theorem 2). But the size of this volume (as a proportion of the total volume) is exponentially small, indicating that it may take exponential time (or other resources) before such a downhill path is found.

Some experiments (or better theory) are needed. The plans for building a machine to carry out such experiments are given in Appendix B. Also, some preliminary results from digital simulations of the machine appear in Appendix C.

3. Programmable Analog Machines

Until now, we have followed the pattern introduced on page 1: Start with an instance P of an NP-complete problem. Build an analog machine based on the specific instance P. Then run the machine to solve P.

A more general approach is worthwhile. In particular, we should be able to design a programmable analog machine which can solve any instance of a given NP-complete problem, up to some fixed size m. The machine will be programmable in the sense that it has relays or other switching devices which are set according to the parameters of a particular instance. *A project: Design and build such a machine for Exact 1-IN-3-SAT, along with an interface to your favorite PC. The PC can use the interface to program the* *machine for any instance of Exact 1-IN-3-SAT up to some fixed size, and read the resulting answer to the programmed instance.*

Appendix A: NP-hardness Proof

Vergis, *et. al.* [1] discuss the following problem, which they call LOC ("Local Optimality Checking"): Given a function $f: \Re^n \to \Re$, a set of feasibility constraints which define a subset $S \subseteq \Re^n$, and an n-dimensional real vector $\mathbf{x} \in S$ — is $\mathbf{x}$ a local maximal optimum for f, within the constraints imposed by S? In order for $\mathbf{x}$ to be a local maximal optimum, there must be a ball centered at $\mathbf{x}$ with some positive radius, such that for every point $\mathbf{y}$ in the ball which is also in S, it follows that $f(\mathbf{y}) \leq f(\mathbf{x})$. Theorem 2 in [1] shows that LOC is NP-hard even if the function f is linear and the constraints S are either piecewise linear or quadratic inequalities.

In this appendix, I'll use a more convenient variation of the LOC problem (more convenient for my applications anyway). In the variation, the question asked is whether the *origin is minimal* (rather than whether an *arbitrary point* is *maximal* as in LOC). Let's call this problem "Local Minimum at Origin" (LoMO), formally defined here:

LoMO: Given a function $f: \Re^n \to \Re$ and a set of feasibility constraints which define a subset $S \subseteq \Re^n$ (such that the origin is in S, and $f(\mathbf{0})=0$), is the origin a local minimal optimum for f with constraints S?

The phrase *"the origin a local minimal optimum for f with constraints S"* means that there is a ball B centered at the origin with some positive radius, such that the value of f is non-negative everywhere in $S \cap B$.

Both LOC and LoMO simplified versions of the widely studied OPTIMIZATION problem, stated here:

OPTIMIZATION: Given a function $f: \Re^n \to \Re$ and a set of feasibility constraints which define a subset $S \subseteq \Re^n$, determine a point $\mathbf{x} \in S$ such that for every $\mathbf{y} \in S$, $f(\mathbf{x}) \leq f(\mathbf{y})$ (or determine that S is empty).

Both LoMO and OPTIMIZATION are NP-hard for a number of combinations of simple functions and constraints, as shown in this table:

The function *f*	**The constraints**	**Comments**
Linear	A set of quadratic inequalities, all of which must be satisfied.	LoMO is shown NP-hard with essentially the same proof as LOC in [1]. This version of OPTIMIZATION was shown NP-hard by Sahni [7].
Quadratic	A set of linear inequalities, all of which must be satisfied.	LoMO is NP-hard. Sahni [7] also showed that this version of OPTIMIZATION is NP-hard. However, if the quadratic function is convex (so that the square matrix of coefficients is positive semi-definite) then a polynomial algorithms exist for OPTIMIZATION and probably for LoMO too [8,9].
Fourth-order polynomial	No constraints	LoMO and OPTIMIZATION are both NP-hard. See [6].
Continuous piecewise-quadratic function with continuous partial first derivatives.	No constraints	This is the version of LoMO used in this report. The problem is NP-hard, as shown in this appendix.

In each case, the coefficients of the function *f* and the constraints may be bounded. This is important since it means that LoMO is *strongly* NP-hard (*i.e.,* all of the numbers used in specifying an instance of LoMO are bounded by a polynomial in the size of the instance [2]).

This appendix demonstrates the NP-hardness of LoMO for the case described in the fourth row of the table. In this case, the function *f* is a continuous piecewise-quadratic function with continuous partial first derivatives, and there are no constraints.

The NP-hardness proof is a transformation from a known variation of 3-satisfyability. The variation, called "Exact 1-in-3-Sat," was shown to be NP-complete in [3], and is defined here:

> Let *V* be a set of Boolean variables. A *3-clause* is any set of three of these Boolean variables. An *assignment* for *V* is a function $A{:}V \to \{true, false\}$. For an assignment *A* and a variable *u*, we say that *u is a true variable* if *A(u)* = *true;* otherwise *u* is a false variable.
>
> Exact 1-in-3-Sat: Given a finite set *C* of 3-clauses, does there exist an assignment such that each of the 3-clauses contains *exactly* one true variable?

When an instance of Exact 1-in-3-Sat has an assignment with exactly one true literal in each clause, then the assignment is called *valid.* We'll now use Exact 1-in-3-Sat to prove the following:

Theorem 3: LoMO is NP-hard, even if the function *f* is a continuous piecewise-quadratic function with continuous partial first derivatives, and there are no constraints.

Proof: Let *P* be an instance of Exact 1-in-3-Sat, and let $v_1, v_2, \ldots, v_m$ be the variables occurring in *P* (with *m* a positive integer). We will define a function $f_P{:}\Re^n \to \Re$, where $n = m + 1$. The function will meet the requirements of the theorem, and have the following property:

The function f_P has a local minimal optimum at the origin

if and only if (†)

The instance *P* has no valid assignment.

The definition of *f* is in terms of *n* real-valued variables. For the first *m* variables of $\Re^n$, we will just use the names $v_1, v_2, \ldots, v_m$, which are the same names as the (boolean) variables in the instance *P.* The *m*+1st (real) variable will be named *w.* Now we can define f_P as the sum of three sorts of terms:

(a) For each variable $x \in V$ the function f_P has one term $W(w, x)$, where W is the continuous piecewise-quadratic function defined by:

$$W(w,x) = \frac{w^2}{2} - \frac{x^2}{4} + \left[\text{if } (w - x \leq 0) \text{ then } \frac{(w-x)^2}{2} \text{ else } 0\right] + \left[\text{if } (w + x \leq 0) \text{ then } \frac{(w+x)^2}{2} \text{ else } 0\right]$$

Figure 7 clarifies the definition of $W(w,x)$. The name W is used because the shape of the function's graph for a fixed positive value of w resembles the letter W, as shown in Figure 7(b). Notice that W and the partial first derivatives $\partial W/\partial x$ and $\partial W/\partial w$ are all continuous.

(b) For each 3-clause $\{x, y, z\}$ that occurs in P, the function f_P has one term:

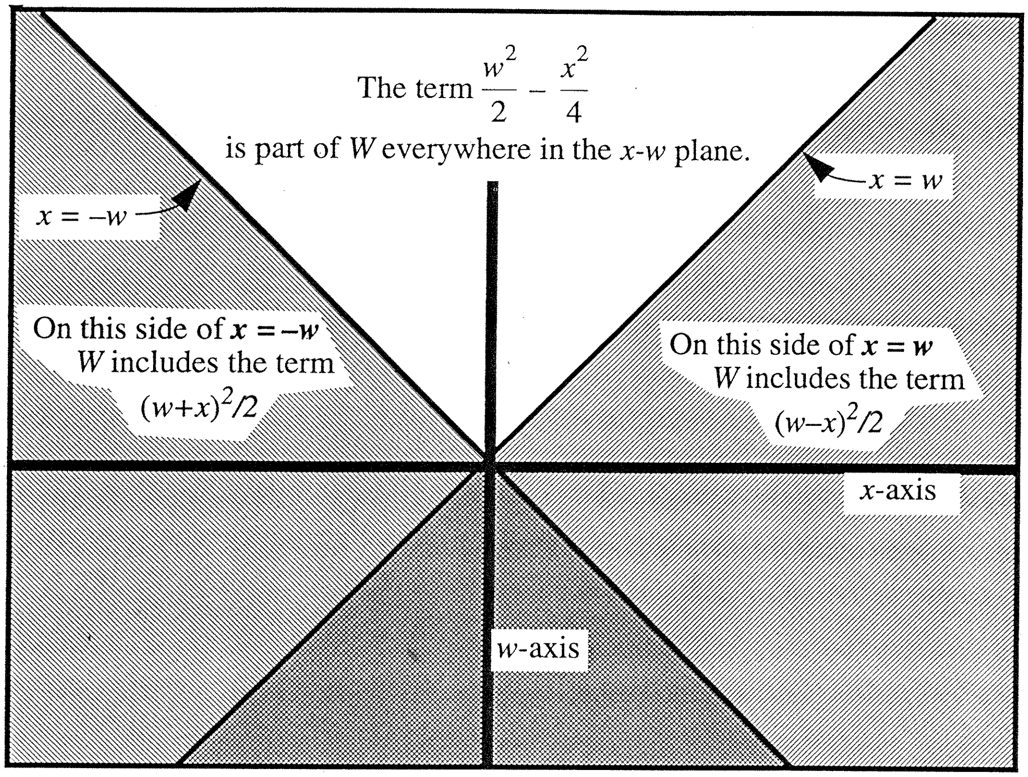
$$(x + y + z + 2w)^2$$

(c) Finally, the function f_P has a single term:

$$-\frac{w^2}{5}$$

Each of the (b) and (c) terms is a quadratic term. The (a)-terms are continuous piecewise-quadratic terms with continuous partial first derivatives. It remains to show that the claim (†) is valid.

For the first half of the claim, assume that P has a valid assignment. We must show that $f(\mathbf{0})$ is not a minimal optimum. For this purpose, we define a vector $\mathbf{x} \in \Re^n$ as follows: The value of the n^{th} component (*i.e.*, w) may be chosen as any positive value ε. For any other component, x, assign 2ε to this component if the corresponding boolean variable x is true in the valid assignment, otherwise assign -2ε to this component of the vector $\mathbf{x}$.



The term $\frac{w^2}{2} - \frac{x^2}{4}$
is part of W everywhere in the x-w plane.
$x = -w$
$x = w$
On this side of $\boldsymbol{x = -w}$ W includes the term $(w+x)^2/2$
On this side of $\boldsymbol{x = w}$ W includes the term $(w-x)^2/2$
x-axis
w-axis

Figure 7(a).

The value of $W(w,x)$ drawn in the x-w plane.

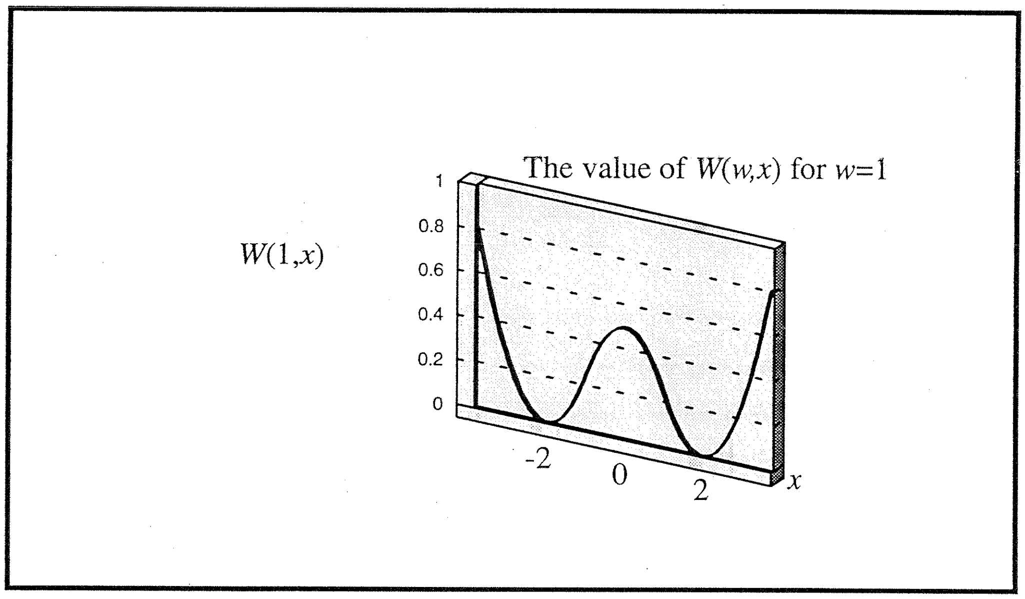


Figure 7(b).

The value of $W(w,x)$ for w=1.

With this definition of $\mathbf{x}$, consider the value of $f_P(\mathbf{x})$. Each of the (a)-terms in $f(\mathbf{x})$ has the form $(w - x/2)^2$ (if $x = 2w$) or has the form $(w + x/2)^2$ (if $x = -2w$). In either case, the (a)-term is zero. Each of the (b)-terms has the form $(x + y + z + 2w)^2$, where exactly one of x, y and z correspond to a true variable. Therefore exactly one of x, y and z is equal to $2w$, and the other two are $-2w$. Therefore each (b)-term is zero. The only remaining term is the (c)-term $-w^2/5$, which is negative, therefore $f_P(\mathbf{x})$ is negative. By making ε sufficiently small, we can make $\mathbf{x}$ as close to $\mathbf{0}$ as we like (keeping $f(\mathbf{x})$ negative) and therefore $f_P(\mathbf{0})$ (which is zero) is not a minimum.

For the second half of the proof, assume that $f_P(\mathbf{0})$ is not a minimal optimum. We must show that P has a valid assignment. For this purpose, let $\mathbf{x}$ be a nonzero vector such that $f_P(\mathbf{x}) \leq 0$. Such a vector exists since $f_P(\mathbf{0})$ (which is zero) is not minimal, therefore there are non-zero points where f_P is non-positive. We note these facts:

(1) The w-component of $\mathbf{x}$ is positive. For if it is zero, then all the terms of $f_P(\mathbf{x})$ are non-negative, and at least one (a)-term (with a non-zero x) is positive, so that $f_P(\mathbf{x})>0$. And if w is negative, then each of the (a)-terms is at least w^2, which is larger than the only possible negative term (the (c)-term), so once again $f_P(\mathbf{x})>0$.

(2) Each component x of $\mathbf{x}$ satisfies:

$$w < |x| < 3w$$

To prove the first inequality, suppose to the contrary that some x has a magnitude $|x| \leq w$, and examine the (a)-term $W(w,x)$. Since w is positive, we lie in the unshaded portion of Figure 7(a) so that:

$$W(w, x) = \frac{w^2}{2} - \frac{x^2}{4} \geq \frac{w^2}{2} - \frac{w^2}{4} = \frac{w^2}{4}$$

Hence, $W(w,x)$ is larger than the magnitude of the only possible negative term (the (c)-term) and $f_P(\mathbf{x}) > 0$. By this contradiction, $w < |x|$.

To prove the second inequality, suppose to the contrary that some x has a magnitude $|x| \geq 3w$. We'll look at the case where x is non-negative (the non-positive case is symmetric). In this case, examine the (a)-term $W(w,x)$:

$$W(w, x) = \left(w - \frac{x}{2}\right)^2 \geq \left(w - \frac{3w}{2}\right)^2 = \left(-\frac{w}{2}\right)^2 = \frac{w^2}{4}$$

Hence, $W(w,x)$ is larger than the magnitude of the only possible negative term (the (c)-term) and $f_P(\mathbf{X}) > 0$.

Now, facts (1) and (2) indicate that none of the components of $\mathbf{X}$ are zero, so we may unambiguously define an assignment for P as follows: if a component x in $\mathbf{X}$ is positive then assign *true* to the boolean variable x. Otherwise assign *false* to the boolean variable x.

It remains to show that this assignment is valid, *i.e.,* that each 3-clause of P has exactly one true literal. We do this in the next two paragraphs.

First note that a clause $\{x,y,z\}$ must have at least one true variable. Otherwise, if all the variables are false, then in the (b)-term $(x + y + z + 2w)^2$, each of x, y and z is negative and moreover (from fact 2) each of x, y and z is smaller than $-w$. Therefore:

$$(x + y + z + 2w)^2 > (-w - w - w + 2w)^2 = w^2$$

Hence, this (b)-term is larger than the magnitude of the only possible negative term (the (c)-term) and $f_P(\mathbf{X}) > 0$.

Next, note that a clause {x,y,z} may not have two or three true variables. Otherwise, if two or three variables are true, then in the (b)-term $(x + y + z + 2w)^2$, two of x, y and z must be positive, and the third might possibly be negative. Without loss of generality, we may assume that it is x which might be negative, so that y and z are both positive. Then (by fact 2) we know that $x > -3w$, and (by fact 2) we know that both y and z are greater than w. Therefore:

$$(x + y + z + 2w)^2 > (-3w + w + w + 2w)^2 = w^2$$

Hence, this (b)-term is larger than the magnitude of the only possible negative term (the (c)-term) and $f_P(\mathbf{X}) > 0$.

Therefore, each 3-clause has exactly one true variable, and the assignment is valid. ❑

Appendix B: Nuts and Bolts for the f_P Machines

This appendix provides the details of the "black box" in Figure 1 for computing the partial first derivatives of the function f_P which was derived from an instance P of Exact 1-in-3-Sat in Appendix A of this report. Throughout this appendix, P will be a fixed instance of Exact 1-in-3-Sat.

The Partial First Derivative $\partial f_P/\partial w$

The function $\partial f_P/\partial w{:}\mathfrak{R}^n \rightarrow \mathfrak{R}$ is a piecewise linear function defined as the sum of the following terms:

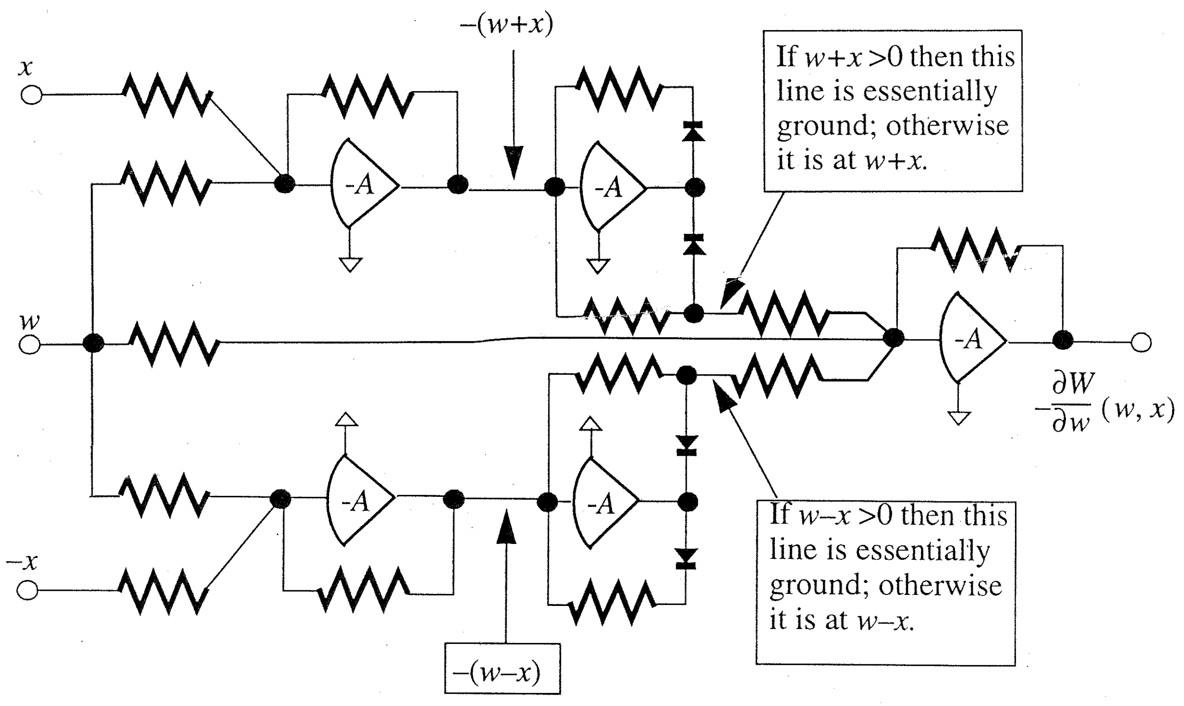
(a) For each variable $x \in V$ the function $\partial f_P/\partial w$ has one term $\partial W/\partial w\,(w, x)$, where W is the continuous piecewise-quadratic function defined in Figure 7 of Appendix A. The value of this term is:

$$\partial W/\partial w\,(w, x) = \begin{array}{c} w \\ + \\ [\text{if } (w - x \le 0) \text{ then } w - x \text{ else } 0] \\ + \\ [\text{if } (w + x \le 0) \text{ then } w + x \text{ else } 0] \end{array}$$

The electronic circuit which computes $-\partial W/\partial w$, as shown in Figure 8. The configuration of the two middle op amps is based on Figure 10-5 from [4]. The other three op amps are functioning as voltage adders.

(b) For each 3-clause $\{x,y,z\}$ the function $\partial f_P/\partial w$ has the linear term $4(x+y+z+2w)$, which can be computed with a voltage adder.

(c) The function $\partial f_P/\partial w$ also contains the linear term $-2w/5$, which can be computed with a constant multiplier.



$-(w+x)$
x
If $w+x>0$ then this line is essentially ground; otherwise it is at $w+x$.
-A
w
$-\frac{\partial W}{\partial w}(w,x)$
If $w-x>0$ then this line is essentially ground; otherwise it is at $w-x$.
$-x$
$-(w-x)$

All resistors have equal resistance.

Figure 8. Analog machine to compute $-\frac{\partial W}{\partial w}(w, x)$

The Partial First Derivatives $\partial f_P/\partial x$ for each $x \neq w$

For any $x \neq w$, the function $\partial f_P/\partial x: \Re^n \to \Re$ is a piecewise linear function defined as the sum of the following terms:

(a) The function $\partial f_P/\partial x$ has one term $\partial W/\partial x\,(w, x)$, where W is the continuous piecewise-quadratic function defined in Figure 7 of Appendix A. The value of this term is:

$$\partial W/\partial x\,(w,x) \quad = \quad \begin{array}{c} -x/2 \\ + \\ [\text{if } (w-x \le 0) \text{ then } x-w \text{ else } 0] \\ + \\ [\text{if } (w+x \le 0) \text{ then } w+x \text{ else } 0] \end{array}$$

The circuit to compute this function is similar to Figure 8.

((b) For each 3-clause $\{x,y,z\}$ which contains x, the function $\partial f_P/\partial x$ contains the linear term:

$$2\,(x + y + z + 2w)$$

Appendix C: Preliminary Digital Simulations

I have performed a few digital simulations of the analog machine which solves Exact 1-in-3-Sat. I actually ran two different versions of the machine: The first version is as shown in Figure 1, keeping track of the position, speed and acceleration of the ball in all n dimensions. The second version simply maintained the velocity of the particle to always be in the direction of steepest descent (*i.e.,* in the direction of the negative gradient).

I ran both simulated machines on fifty randomly-generated satisfyable instances of Exact 1-in-3-Sat, with ten variables and twenty clauses each. Each simulation ran for 100 seconds of simulated time, with an initial position so that each coordinate was randomly selected with a maximum magnitude of 10^{-7}. Time was simulated in discrete slices of 10^{-4} seconds. The results are summarized in this table:

50 Simulated Instances	**% of Instances Solved within 100 simulated seconds**	**Average time spent on a solved instance**	**Maximum time spent on a solved instance**
First Version	82%	29.2 sec	99.6 sec
Second Version	30%	0.0029 sec	0.0122 sec

The second version certainly works quickly – when it works, that is. When the second version failed, the state space of the machine moved close enough to the origin that the gradient was too flat to make continued progress. Perhaps the second method could solve a higher percentage of instances if it were allowed to restart at a new random starting position whenever it approaches the origin too closely.

And, of course, I am uncertain how closely the digital simulations models the real thing.

Notes Added June 1997

I have built a small version of the machine to solve the case of Exact 1-in-3-Sat with three Boolean variables and one clause that contains these three variables. The machine correctly solves the problem, moving away from the origin in a direction that pulls one of the variables positive (true) and the other two negative (false). There are three such directions and the machine seems to choose between the directions nondeterministically (although not equally likely). I don't know what causes one direction to be more likely than the others, but it's probably small differences in the capacitors or other components.

I'm building a programmable version of the machine this summer, capable of handling any instance with up to 16 variables and up to 20 clauses. The programming is done with patch cords and I hope to report on the results later this year. The dynamical system that I'm using has one change from the idea in this report: I am starting with the function f_P described in Appendix A, but I am using $-\nabla f_P(\mathbf{x})$ as the instantaneous velocity of the dynamical system (rather than the instantaneous acceleration). This cuts the number of itegrators in half.

References

[1] Anastasios Vergis, Kenneth Steiglitz and Bradley Dickinson. "The Complexity of Analog Computation", *Mathematics and Computers in Simulation* 28 (1986) 91-113.

[2] Michael R. Garey and David S. Johnson. *Computers and Intractability, A Guide to the Theory of NP-Completeness.* W.H. Freeman and Company (San Francisco, 1979).

[3] T.J. Schaefer. "The complexity of satisfyability problems", *Proc. 10th Ann. ACM Symp on Theory of Computing.* Association for Computing Machinery (New York, 1978), pp. 216-226. Note that the 1-IN-3_SAT problem remains NP-complete even if negated literals are forbidden.

[4] Michael G. Rekoff, Jr. *Analog Computer Programming.* Charles E. Merrill Books, Inc. (Columbus, Ohio, 1967).

[5]Yinyu Ye. "An extension of Karmarkar's algorithm and the trust region method for quadratic programming," in *Progress in Mathematical Program, Interior-Point and Related Methods,* edited by Nimrod Megiddo, (Springer-Verlag, New York, 1989).

[6] L. Blum, M. Shub and S. Smale. "On a theory of computation and complexity over the real numbers: NP completeness, recursive functions and universal machines", *Bull. Amer. Math. Soc.* **21** (1989), 1-46.

[7] S. Sahni. "Computationally related problems," *SIAM J. Computing* **3** (1974), 262-279.

[8] M.K. Kozlov, S.P. Tarasov and L.G. Hač ijan. "Polynomial solvability of convex quadratic programming," *Dokl. Akad. Nauk SSSR* **248** (1979), 1049-1051. Translated to English in *Soviet Math Dokl.* **20**, 1108-1111.