GLOBAL OPTIMIZATION: A NAIVE APPROACH

by

Leon O. Chua

Memorandum No. UCB/ERL M89/118

15 August 1989

GLOBAL OPTIMIZATION: A NAIVE APPROACH

by

Leon O. Chua

Memorandum No. UCB/ERL M89/118

15 August 1989

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

GLOBAL OPTIMIZATION: A NAIVE APPROACH

by

Leon O. Chua

Memorandum No. UCB/ERL M89/118

15 August 1989

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

Global Optimization: A Naive Approach †

Leon. O. Chua ‡

This note proposes a unified approach for finding the *global optimum* of a continuously differentiable function $\phi: R^n \rightarrow R$ of n real variables $\mathbf{x} = (x_1, x_2, \cdots, x_n) \in R^n$, for both constrained and unconstrained problems. This approach may be implemented either numerically or in *real time* via an associated *nonlinear programming circuit* to be described at the end of this note.

1. Global Optimization without Constraints

Without loss of generality we will consider only the minimization problem. Let $\mathbf{x}^*$ be a *global minimum* of $\phi(\mathbf{x})$; i.e.,

$$\mathbf{x}^* = \min_{\mathbf{x} \in R^n} \phi(\mathbf{x}) \tag{1}$$

If $\phi(\mathbf{x})$ has multiple local extremum points, most current deterministic optimization algorithms will generally converge to some local minimum [1]. Our global optimization approach consists of transforming the *unconstrained* optimization problem into the following *nonlinear programming problem:*

$$\underset{\mathbf{x} \in R^n}{\textit{minimize}}\ \phi(\mathbf{x}) \tag{2}$$

subject to the *inequality* constraint

$$\phi(\mathbf{x}) \leq \phi(\hat{\mathbf{x}}) - \varepsilon \tag{3}$$

where $\hat{\mathbf{x}}$ is a local minimum; i.e., $\phi(\hat{\mathbf{x}}) > \phi(\mathbf{x}^*)$ and ε is a *small positive* number chosen so that the plane $\phi(\mathbf{x}) = \phi(\hat{\mathbf{x}}) - \varepsilon$ in the ϕ vs. $\mathbf{x}$ space (see Fig. (1)a) lies just below the local minimum $\phi(\hat{\mathbf{x}})$. Solving the above problem by any standard nonlinear programming method yields a new extremum point $\hat{\mathbf{x}}'$ with the property that $\phi(\hat{\mathbf{x}}') < \phi(\hat{\mathbf{x}})$. If $\phi(\mathbf{x})$ has a finite

† This work is supported in part by the Office of Naval Research under grant N00014-89-J-1402 and the National Science Foundation grant MIP-8614000.

‡ L. O. Chua is with the University of California at Berkeley, Berkeley, CA 94720.

number "N" of isolated local extremum points, our approach needs only be repeated at most N times, each time involving a "lower" local minimum. This rather naive strategy can be summarized as follow:

Global Minimization Strategy

Step 0 Choose a small positive number $\varepsilon > 0$ and an initial guess $\hat{\mathbf{x}}^{(0)}$. †

Step 1 At the kth recursion ($k = 0, 1, 2, ..., m \leq N$), let $\hat{\mathbf{x}}^{(k)}$ denote the local minimum calculated from the preceding step. Calculate

$$\phi_0^{(k)} = \phi(\hat{\mathbf{x}}^{(k)}) - \varepsilon$$

Step 2 Solve the constrained minimization problem

$$\underset{\mathbf{x} \varepsilon R^n}{\textit{minimize}}\ \phi(\mathbf{x})$$

subject to the inequality constraint

$$\phi(\mathbf{x}) \leq \phi_0^{(k)}$$

If *step 2* has a solution, call it $\hat{\mathbf{x}}^{(k+1)}$ and return to *step 1* with k updated to $k+1$. Otherwise, $\hat{\mathbf{x}}^{(k)} = \mathbf{x}^*$ is the global minimum.

Note that in general, the global minimum will be found in a significantly smaller number "m" of recursions than the total number N of local extremum points. This is because it is highly unlikely that *step 2* would converge to the *next* lower local minimum. Consequently, all "intermediate" local minima that fall between $\phi(\hat{\mathbf{x}}^{(k)})$ and $\phi(\hat{\mathbf{x}}^{(k+1)})$ are *automatically* eliminated from future considerations.

It is important to observe that while our above strategy seems to be naive if not obvious, it offers the following three novel features:

† In the case of the nonlinear programming circuit [3-5] to be proposed below, the initial noise voltage on the capacitors is the analog of the initial guess.

1. The local minimum calculated from each recursion is guaranteed to be better than the previous one; namely, $\phi(\hat{\mathbf{x}}^{(k+1)})$ is less than $\phi(\hat{\mathbf{x}}^{(k)})$ for all k. This monotone cost-decreasing property differs from other algorithms where the next calculated local minimum could turn out to be worse than before, or one which in fact has been previously obtained.

2. As illustrated below, our strategy can be formulated into a new minimization problem (with or without constraints) so that all previously calculated local minima are automatically suppressed *algebraically*, and not by an algorithm which compares them to the current outcome.

3. Our strategy makes it possible to build a "neural network" [5] capable of obtaining the *global* minimum in *real time*. In fact, the *nonlinear programming circuit* to be proposed below, is our main motivation of this note.

While any standard code for solving nonlinear programming problems can be used to implement the above global optimization strategy, it is possible to transform the above constrained minimization problem *(step 2)* into the following *equivalent* unconstrained minimization problem by augmenting a new equation and a new "slack variable" y; namely, solve the system of $n+1$ nonlinear equations in the $n+1$ unknowns $x_1, x_2, \cdots, x_n, y$

$$\frac{\partial \phi(\mathbf{x})}{\partial x_j} = 0, \qquad j = 1, 2, \cdots, n \tag{4}$$

$$\phi_0^{(k)} - \phi(\mathbf{x}) - g(y) = 0 \tag{5}$$

where $g(y)$ is any continuous "positive" function ($g(y) \geq 0$) from R^1 into R^1. For example, we can choose $g(y) = y^2$ or $g(y) = |y|$ if differentiability is not essential for the numerical algorithms.

Example

To illustrate the basic idea behind our global optimization strategy, let us minimize the function

$$\phi(x) = x^4 - 2x^2 + 0.5x \tag{6}$$

which has 2 local minima at $\hat{x}^{(1)} = 0.9304$ and $\hat{x}^{(2)} = -1.0574$, as shown in Fig. 1(a). Assuming $\hat{x}^{(1)}$ has been found, our goal is to find another local minimum $\hat{x}^{(2)}$ so that $\phi(\hat{x}^{(2)}) < \phi(\hat{x}^{(1)}) = -0.5167$. Choosing $\varepsilon = 0.1083$, we obtain $\phi_0^{(1)} = -0.625$. This is shown in Fig. 1(a) by the horizontal dash line which falls below the local minimum $\phi(\hat{x}^{(1)})$ but above the next lower minimum $\phi(\hat{x}^{(2)})$, which in this example, is the global minimum†. The inequality constraint

$$\phi(x) = x^4 - 2x^2 + 0.5x \leq -0.625 \tag{7}$$

defines the feasible region $x_b \leq x \leq x_a$, where $x_a = -0.4685$ and $x_b = -1.4297$.

Applying (4) and (5) with $g(y) = y^2$, the next local minimum is found by solving the system of 2 nonlinear equations:

$$\phi'(x) = 4x^3 - 4x + 0.5 = 0 \tag{8}$$

$$-0.625 - x^4 + 2x^2 - 0.5x - y^2 = 0 \tag{9}$$

Solving (8) and (9) by the Newton-Raphson method, we find the iteration converges to the solution $x^{(2)} = -1.0574$ for all initial guess $x \leq -0.57735$ and $y \neq 0$.

Our next recursion with $x^{(k)} = \hat{x}^{(2)}$ lead to divergence for all initial guesses. This is taken as our *stopping rule* and an indication that the solution $\hat{x}^{(2)} = -1.0574$ is in fact a global minimum. The loci of $\phi'(x) = 4x^3 - 4x + 0.5$ is plotted in Fig. 1(b) as a function of x. Below it is a plot of $y = \pm\sqrt{-0.625 - \phi(x)}$, which shows the feasible region of x is bounded by a symmetrical closed loop. Note that among the 3 projections through the 3 roots of $\phi'(x) = 0$ at $x = 0.9304$, 0.1270 and -1.0574, only the last one intersects the feasible region, as expected.

The same result can be obtained by replacing (5) with many other equations, some possibly having better convergence properties. For example, we can replace (5) by (18) (to be derived in the following section) with $f_i(x) = \phi_0^{(k)} - \phi(x)$; namely,

$$y - \phi_0^{(1)} + \phi(x) + |\, y + \phi_0^{(1)} - \phi(x)\,| = 0 \tag{10}$$

The loci of (10) in the x-y plane is defined by (20) - (22), in view of the *theorem* proved in

† To avoid clutter in the graph in Fig. 1, we have chosen first $\phi_0^{(1)} = -0.625$, thereby deducing that $\varepsilon = 0.1083$. In practice, ε would be an arbitrarily chosen positive number, whose value is small relative to the numerical scale of $\phi(x)$.

Section 2, and corresponds to the inverted rectangular well shown in Fig. 1(d). Again, only $x = x^{(2)}$ intersects this well. Our Newton-Raphson iteration of (8) and (10) reveals a faster convergence property comparing to that of Eqs. (8) and (9).

2. Global Optimization with Constraints

Consider next the constrained optimization problem

$$\underset{\mathbf{x}\varepsilon R^n}{minimize}\ \phi(\mathbf{x}) \tag{11}$$

subject to "p" *inequality* constraints

$$\mathbf{f}_i(\mathbf{x}) \geq 0, \qquad i = 1, 2, \cdots, p \tag{12}$$

and "q" *equality* constraints

$$\mathbf{f}_i(\mathbf{x}) = 0, \qquad i = p+1, p+2, \cdots, p+q \tag{13}$$

Our global optimization strategy presented above is also applicable, *mutatis mutandis*, to this problem, provided (12) and (13) are introduced as *additional* constraints. In this case, $\hat{\mathbf{x}}^{(k)}$ is a *local* minimum only if the constraints (12) are *inactive*, i.e., $\mathbf{f}_i(\mathbf{x}) > 0$, $i = 1, 2, \cdots, p$.

Just as in *Section 1*, we can transform the resulting *nonlinear programming* problem into an equivalent *unconstrained* minimization problem. However, instead of using (5), which would not satisfy the Kuhn-Tucker necessary condition [2] for the "augmented" constraint (12), we propose the following equivalent system of $(n+p+1)$ equations in the $(n+p+1)$ unknowns $(x_1, x_2, \cdots, x_n, y_0, y_1, y_2, \cdots, y_p)$:

$$\frac{\partial \phi(\mathbf{x})}{\partial x_j} + \sum_{i=0}^{p} y_i \frac{\partial \mathbf{f}_i(\mathbf{x})}{\partial x_j} + \sum_{i=p+1}^{p+q} y_i \frac{\partial \mathbf{f}_i(\mathbf{x})}{\partial x_j} = 0, \qquad j = 1, 2, \cdots, n \tag{14}$$

$$\mathbf{g}(\mathbf{h}(y_i) + \mathbf{f}_i(\mathbf{x})) - y_i = 0, \qquad i = 0, 1, 2, \cdots, p \tag{15}$$

where†

$$\mathbf{f}_0(\mathbf{x}) \equiv \phi_0^{(k)} - \phi(\mathbf{x}) \tag{16}$$

$\mathbf{h}: R^1 \rightarrow R^1$ is any continuous *strictly monotone increasing* function satisfying $\mathbf{h}(0) = 0$, and

$$\mathbf{g}(z) = \begin{cases} 0, & z \geq 0 \\ \mathbf{h}^{-1}(z), & z < 0 \end{cases} \tag{17}$$

Many choices of $\mathbf{h}(\cdot)$ and $\mathbf{g}(\cdot)$ are available. Two simple choices are:

Example 1. Choose $\mathbf{h}(z) = z$. In this case, $\mathbf{g}(z) = \frac{1}{2}(z - |z|)$ and (15) becomes

$$y_i - \mathbf{f}_i(\mathbf{x}) + |y_i + \mathbf{f}_i(\mathbf{x})| = 0 \qquad i = 0, 1, 2, \cdots, p \tag{18}$$

Example 2. Choose $\mathbf{h}(z) = z^{\frac{1}{3}}$. Hence, $\mathbf{g}(z) = \begin{cases} 0, & z \geq 0 \\ z^3, & z < 0. \end{cases}$ In this case, (15) becomes

$$\mathbf{g}(y_i^{\frac{1}{3}} + \mathbf{f}_i(\mathbf{x})) - y_i = 0, \qquad i = 0, 1, 2, \cdots, p \tag{19}$$

The proof that the system (14) - (15) is indeed equivalent to the nonlinear programming problem (11) - (13) follows immediately from the following fundamental result:

Theorem

The set of points $(\mathbf{x}, y_i)$ satisfying (15) is defined by the following constraints:

$$\mathbf{f}_i(\mathbf{x}) \geq 0 \tag{20}$$

$$y_i \leq 0 \tag{21}$$

† The superscript "k" in $\phi_0^{(k)}$ corresponds to the kth recursion in the global optimization strategy in *Section 1.*

$$y_i \, \mathbf{f}_i(\mathbf{x}) = 0 \tag{22}$$

Proof. It suffices to consider 2 mutually exclusive cases:

(a).

$$z \equiv \mathbf{h}(y_i) + \mathbf{f}_i(\mathbf{x}) \geq 0 \tag{23}$$

In this case, (17) implies $\mathbf{g}(z) = 0$. It follows from (15) that $y_i = 0$. Hence, $y_i \, \mathbf{f}_i(\mathbf{x}) = 0$ and $\mathbf{h}(y_i) = 0$. Consequently, (23) implies $\mathbf{f}_i(\mathbf{x}) \geq 0$.

(b).

$$z \equiv \mathbf{h}(y_i) + \mathbf{f}_i(\mathbf{x}) < 0 \tag{24}$$

In this case, (15) and (17) imply that

$$\mathbf{h}^{-1}(\mathbf{h}(y_i) + \mathbf{f}_i(\mathbf{x})) - y_i = 0 \tag{25}$$

which we can recast as

$$\mathbf{h}(y_i) + \mathbf{f}_i(\mathbf{x}) = \mathbf{h}(y_i) \tag{26}$$

Equation (26) implies $\mathbf{f}_i(\mathbf{x}) = 0$ and hence $y_i \, \mathbf{f}_i(\mathbf{x}) = 0$. Now since $\mathbf{h}(\cdot)$ is a strictly monotone increasing function, Eq. (17) implies that $\mathbf{g}(z) \leq 0$. It follows from (15) that $y_i \leq 0$. □

Assuming that the inequality constraints (12) satisfy some appropriate *constraint qualifications* [2], which is impractical to check but is almost always satisfied in practical problems, the preceding theorem guarantees that (14) and (17) satisfy the "generalized Kuhn-Tucker necessary conditions" [2] for solving the nonlinear programming problem (11) - (13).

Simple 2-dimensional numerical examples have shown that the above *global optimization strategy* to be quite efficient and robust. Even the non-differentiable nature of the absolute value function in (18) did not seem to pose a problem when the Newton-Raphson method is used to solve the several unconstrained minimization problems we have tried. This robustness can be explained in part by the nature of our strategy where the global optimum can never occur at the boundary of the constraint (3), and hence the non-differentiable point is never attained during iteration. Needless to say, many more examples involving a large number of variables and constraints must be tested before the general applicability and efficiency of our global optimization strategy can be ascertained. Such results will be reported in a future

publication.

For engineers unfamiliar with the theory of nonlinear programming, our global optimization strategy can be implemented by a general purpose nonlinear circuit simulation program, such as some versions of SPICE, by simulating an associated "*dc*" *nonlinear programming circuit* as described in [3]. To use this approach, one simply incorporates the *inequality 3* as an additional constraint, and updates it with $\phi_0^{(k)}$ as each new local minimum is calculated. The resulting circuit is *purely* resistive and can be solved any nonlinear resistive circuit simulation program which accepts the types of nonlinear characteristics and controlled sources present in the circuit.

Rather than simulating "N" updated "dc" nonlinear resistive circuits, an alternate "cookbook" approach is to simulate "N" *dynamic nonlinear programming circuits* [4] derived from the resistive nonlinear programming circuit in [3] by connecting a parasitic capacitor across each output voltage node and the ground. The resulting dynamic circuit can be easily proved to be *completely stable* [5], and hence the *transient* response from any initial condition must converge to a local equilibrium point, which coincides with a local minimum, subject to the imposed constraints.

In carrying out the above "cookbook" approach via *dc* or *transient* circuit simulations, the objective function $\phi(\mathbf{x})$ and the inequality constraints are assumed to be specified in *analytic* form, so that their partial derivatives are available either in explicit form, or may be calculated by some built-in *symbolic* differentiation software, as in INSITE [6]. In practice, $\phi(\mathbf{x})$ often represents some "cost function" (e.g., power dissipation, switching speed, etc.) associated with some circuit to be minimized, and is therefore not available in *analytic* form. Rather, $\phi(\mathbf{x})$ must be calculated at each iteration by first solving the circuit's nonlinear equations. In this case, $\phi(\mathbf{x})$ is defined only *implicitly*, and the *gradient* information (i.e., the partial derivatives) needed in the nonlinear programming circuit may be obtained by numerical differentiation or by a more efficient *variational approach* such as the *adjacent network concept* [7].

The most intriguing application of our global optimization strategy, however, is the possibility of obtaining the *global optimum* in *real time* by actually *building* (not simulating) the nonlinear simulating circuit, as described in [4-5] which incorporates the additional inequality constraints (3). In this case, even the updating of this constraint can also be achieved in *real time* with a high-speed sample-and-hold type controlling circuitry for automatically updating the "threshold" voltage source associated with $\phi_0^{(k)}$ whenever the transient decays sufficiently

close to a local equilibrium. Such a global nonlinear programming circuit should be specially useful in applications (such as robotics) where real time global optimization is essential.

For *arbitrary* nonlinearities, such a VLSI circuit may not be feasible using current technology. However, for large-volume applications, a dedicated VLSI circuit can now be built to solve many *standard* nonlinear programming problems. (e.g., linear and quadratic programming problems) in real time. As new technology advances, even more general "programmable" nonlinear programming neural networks may eventually be feasible.

Reference

[1] L. C. W. Dixon and G. P. Szego (editor), *Towards Global Optimization* 2, North Holland Publishing Company, Amsterdam, 1978

[2] M. Avriel, *Nonlinear Programming*, Prentice Hall, Englewood, New Jersey, 1976

[3] L. O. Chua and G. N. Lin, "Nonlinear Optimization with Constraints: a Cookbook Approach," *International Journal of Circuits Theory and Applications*, Vol. 11, pp. 141-159, 1983.

[4] L. O. Chua and G. N. Lin, "Nonlinear Optimization with Constraints," *IEEE Trans. on Circuits and Systems*, Vol. CAS-31, pp 182-188, 1984.

[5] M. Peter Kennedy and L. O. Chua, "Nonlinear Networks for Nonlinear Programming," *IEEE Trans. on Circuits and Systems*, Vol. 35, pp. 554-562, May, 1988.

[6] T. S. Parker and L. O. Chua, *Practical Numerical Algorithms for Chaotic Systems*, Springer-Verlag, New York, N.Y., 1989.

[7] L. O. Chua and P. M. Lin, *Computer-Aided Analysis of Electric Circuits: Algorithms and Computational Techniques*, Prentice Hall, Englewood Cliff, New Jersey, 1975.

Figure Caption

Fig. 1 An example for illustrating the geometrical idea behind our global optimization algorithms.

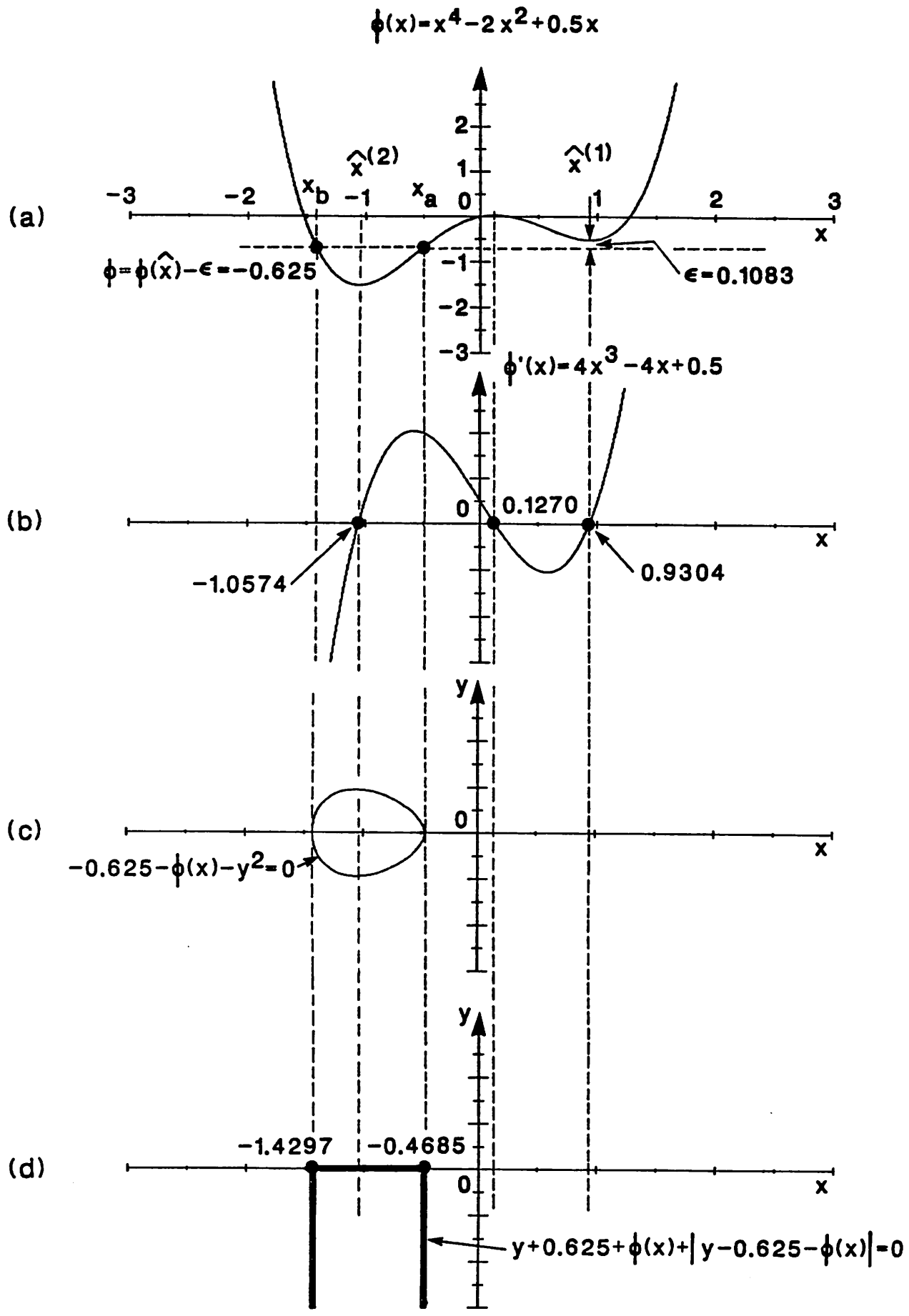
$\phi(x)=x^4-2x^2+0.5x$
(a)
$\hat{x}^{(2)}$
$\hat{x}^{(1)}$
x_b
x_a
$\phi=\phi(\hat{x})-\epsilon=-0.625$
$\epsilon=0.1083$
$\phi'(x)=4x^3-4x+0.5$
(b)
0.1270
−1.0574
0.9304
(c)
$-0.625-\phi(x)-y^2=0$
(d)
−1.4297
−0.4685
$y+0.625+\phi(x)+|y-0.625-\phi(x)|=0$
x
y

Figure 1