Determination of Exponential Parameters

K. Irene Snyder

Wesley E. Snyder

Center for Communications and Signal Processing
Department Electrical and Computer Engineering
North Carolina State University

TR-91/11
August 1991

Determination of Exponential Parameters

K. Irene Snyder
Department of Psychology
University of North Carolina at Chapel Hill

Wesley E. Snyder
Dept. of Radiology
Bowman Gray School of Medicine
Winston-Salem, NC

The problem of finding the parameters of exponential processes is discussed. Difficulties with numerical approaches based on gradient descent are illustrated. A new method for finding those parameters is presented, which requires virtually no manual intervention, and which can find globally optimal estimates

1 Introduction

Researchers in perception and psychophysics often find it necessary to plot data points or measurements in logarithmic curves against time. Displacement threshold determination[31] , tilt aftereffect growth and decay[12,21] contrast and light adaptation [37], vibrotactile sensitivity [15], and many others produce data that is best fit by logarithmic curves. Stevenson's power law and Atkinson's more general equation[2] show a generally exponential/logarithmic relationship between sensation magnitude and stimulus intensity.

Quantification of the physiological processes underlying the observed behavior often requires determining the parameters of an exponential function; however, a difficulty lies in determining the best fitting curve for data on an exponential or logarithmic curve, since data unfortunately rarely lies precisely on a simple exponential curve. In this paper, we address the fundamental problem of finding the best estimate of the parameters of an exponential-type functions, given noisy measurements of that function. The resulting technique is demonstrated here for simple exponential functions, but is generally applicable to more complex functions such as those described by Atkinson[2]. Given an exponential function such as

$$y = a + be^{cx},$$

to find the values of the parameters a, b and c which provide the best fit, we find the curve which minimizes the error, or the distance between the data points and the curve. We describe this error by

$$\sum_i [(y_i - (a + be^{cx_i})]^2 \qquad (1.1)$$

where the sum is taken over a set of measurements. Here, E is referred to as the means squared error.

The easiest way to find the minimum of such an expression is to find the zeroes of the derivatives, but differentiating this equation gives us three derivatives:

$$\frac{\partial E}{\partial a} = -2\sum 1$$

$$\frac{\partial E}{\partial b} = -2\sum e^{cx_i} \qquad (1.2)$$

$$\frac{\partial E}{\partial c} = -2b\sum e^{cx_i} x_i$$

An algebraic solution to this simultaneous system of equations is intractable. Thus, we are forced to consider numerical methods.

1.1 Gradient Descent

Gradient descent, a standard numerical technique for optimization, finds the minimum by stepping slowly down the slope since the slope always points away from the minimum (or at least, from *some* minimum). Given some scalar function $E(x)$, and a current estimate (at iteration k) of $x^{(k)}$ to find the downhill point $x^{(k+1)}$ gradient descent uses

$$x^{(k+1)} = x^{(k)} - \frac{E'(x^{(k)})}{|E''(x^{(k)})|}, \qquad (1.3)$$

where E' and E'' are the first and second derivatives of E with respect x.

There are a number of problems inherent in gradient descent-type methods:

- **Parameter sensitivity** The ranges in the parameters, and their sensitivity to perturbations may easily differ by orders of magnitude. For example, the parameter c described in the equations above is critical to the stability of the algorithm, and *very* sensitive to small errors. To compensate for this sensitivity, we divide the gradient by the second derivative (in the scalar case, or by some norm of the Hessian matrix for more complex problems). While use of the second derivatives solves some of the sensitivity problem, it introduces a second problem: algebraic tedia.
- **Representational complexity** To perform gradient descent, one must analytically evaluate first and second partial derivatives. This algebraic process, while straightforward, is tedious and prone to error. Fortunately, there are now a number of symbolic math software packages which can be used to automated this process. For the many users who do not have easy access to such tools, the simple process of correctly differentiating complex expressions is tedious and time consuming at best.
- **Plateaus** Exponential functions have a notorious tendency to have plateaus in the corresponding MSE fit functions. For example, Figure 1 illustrates the error measure of Equation (1) plotted vs. c with a and b held at their optimal values. The slope of this curve becomes *arbitrarily* small as one moves to the left along the c axis. If one choose a

stopping criterion for gradient descent such as "stop if the magnitude of the gradient is less than T", it is trivial to find such a point by moving along the c axis *away* from the minimum.

- **Local Minima**
  The primary flaw with gradient descent as a means of solving this type of minimization problem is that (unless it gets caught on a plateau), it finds the minimum nearest to the starting point, which may or may not be the absolute minimum.

1.2 Simulated Annealing

The minimization technique known as simulated annealing [19,1] avoids the problems of local minima and plateaus and may be described as follows:

1. Choose (at random) an initial value of x.
2. Generate a point y which is a "neighbor" of x.
3. If $E(y) < E(x)$, y becomes the new value of x.
4. If $E(y) > E(x)$, compute $P_y = \exp(-E(x)-E(y)/T)$. If $P_y > R$ for R (a random number uniformly distributed between 0 and 1) then accept y.
5. Decrease T slightly.
6. If $T > T_{min}$, go to 2.

In step 3, we perform a descent, so that we always fall "downhill". In step 4, we make it possible to sometimes move uphill and get out of a valley. Initially, we ignore T and note that if y represents an uphill move, the probability of accepting y is proportional to $e^{-(E(y) - E(x))/T}$. Thus, uphill moves *can* occur, but are exponentially less likely to occur as the size of the uphill move becomes larger. The likelihood of an uphill move is, however, strongly influenced by T. If T is very large, all moves will be accepted. As T is gradually reduced, uphill moves become less likely until for low values of T, $T<<(E(y) - E(x))$, such moves essentially cannot occur.

In the case of combinatorial optimization, where all the variables take on only one of a small number (usually two) of possible values, the "neighbor" of a vector x_1 is another vector x_2 such that only one element of x is changed to create x_2. For problems (such as fitting an exponential) where the variables take on continuous values, it is

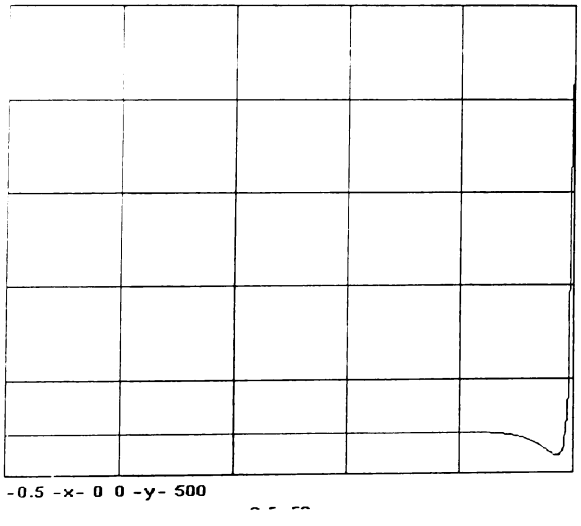

Figure 1.
Fit error vs. exponential parameter c

much more difficult to quantify the concept of "neighbor".

In the next section, we describe a new minimization strategy, which handles continuously-valued variables.

2 Continuous Optimization (tree annealing)

Although some work has been done on extending SA to problems with continuous variables[36], the nature of the SA algorithm makes it best suited for solving problems in combinatorial optimization, in which the variables take on only discrete values. This is primarily due to the difficulty in specifying, for a particular problem, precisely what the "neighborhood" of a continuously-valued variable is. We will not, in this paper, attempt to survey all the other (that is, not based on SA) methods for continuous optimization. See [8] for more information.

In this section, we discuss a method for finding the minimum of functions of continuously-valued variables. We find it convenient to think of the optimization problem as a search: the minimum lies somewhere in a bounded hyperspace of dimension d. It is not practical to use any sort of array structure to store a representation of such a space, since the storage rapidly becomes prohibitive. Instead, we use a tree.

The minimization method described here, which we call "tree annealing" is an extension of the familiar Metropolis [23]algorithm of simulated annealing, but handles continuously valued variables in a natural way.

We assume we are searching for the minimum of some function $H(\boldsymbol{x})$ where the d-dimensional vector $\boldsymbol{x}$ has continuously-valued elements. Furthermore, we assume a bounded search space $S \subseteq \Re^d$, which we will represent with a dynamic data structure.

We use a k-d tree in which each level of the tree represents a binary partition of one particular degree of freedom(DOF). Each node may thus be interpreted as representing a hyperrectangle, and its children therefore represent the smaller hyperrectangles resulting from dividing the parent along one particular DOF. In Figure 2, we illustrate a one-dimensional energy function, and show how a resulting partition tree provides more resolution in the vicinity of minima.

In Figure 3, we illustrate a 2-D example of the partition tree and how nodes in the tree correspond to successive

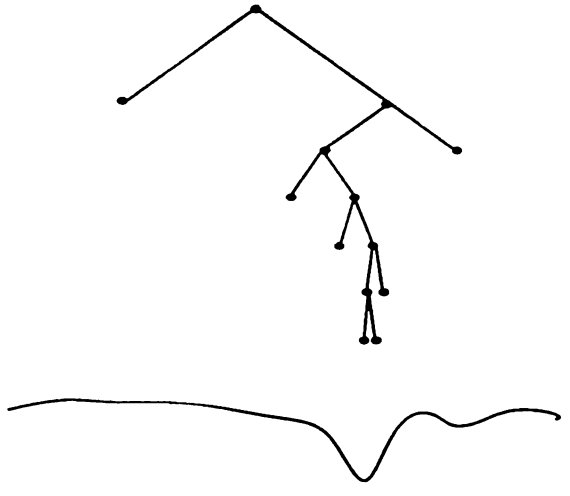

Figure 2. A one dimensional search space and corresponding partition tree It is important to remember that the tree is built using a random process therefore, the tree is *likely* to possess more depth (resolution) in the vicinity of minima. Therefore this figure shows only what the tree is *likely* to be.

refinements.

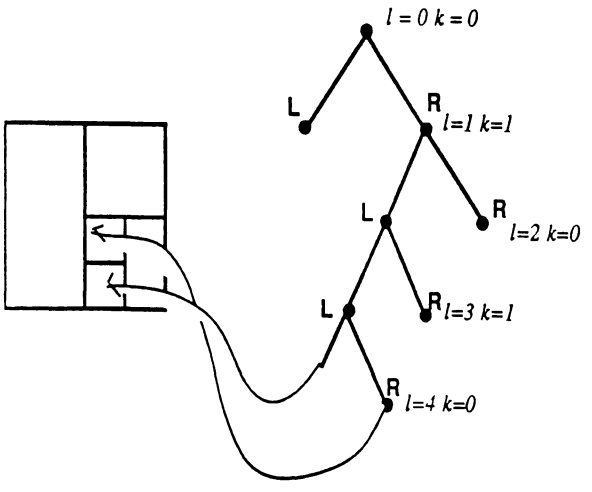

Figure 3. A two dimensonal example

In general, for a d-dimensional problem,

a node at level l will divide the k^{th} degree of freedom in half,

where $k = l \bmod d$ The two children of each node represent the subspace resulting from dividing the parent subtree in half along the k^{th} sample, in a 6 DOF problem, suppose the domain of DOF 4 is $0 \le x_4 \le 1$. Then a node at level 22 represents a partition of DOF 4, a par n whose width is

$$\frac{1}{2^{\lfloor 22/6 \rfloor}} = 0.0625.$$

The metropolis algorithm proceeds as follow

1. Given a current estimate $\mathbf{x} \in S$, generate a "neighboring" point $\mathbf{y} \in S$, with generation probability $g(\mathbf{y} \mid \mathbf{x})$. It is necessary to assume that g is symmetric in the following sense:
$$g(\mathbf{y} \mid \mathbf{x}) = g(\mathbf{x} \mid \mathbf{y}).$$
2. Accept the point $\mathbf{y}$ as the new estimate with probability
$$\min\left(1, \frac{p(\mathbf{y})}{p(\mathbf{x})}\right), \qquad (2.1)$$
where the probabilities are Gibbs, (i.e., with form $p(\lambda) \propto \exp\left(-\frac{H(\lambda)}{T}\right)$. Consider, for a moment, the form of Equation (1). In it, we point out that the decision to accept or reject $\mathbf{y}$ is based on a *probability*. A moment's reflection ll convince the reader that this is equivalent to steps 3 and 4 of the simulated annealing algorithm.

Omitting for the moment discussion of annealing, that is, reducing T, we consider how the Simulated Annealing strategy is modified by this "tree annealing" (TA) algorithm.

2.1 Growing and Searching the Tree

While the tree represents the entire search space, we build it with higher resolution in the vicinity of local minima. To see how this works, let us assume the process has been running for a while, and the tree has already been partially built. At each node, two numbers have been stored, n_L and n_R, which represent how many times in the past that an acceptable point has been found in the left and right subtrees, respectively, of that node.

[1]This description is slightly different from the one presented earlier, since we want to emphasize the importance of the generation process

Following the basic Metropolis schema, we are given a vector **x**, which we refer to as "the current sample" and corresponding objective H(**x**). To choose a candidate point:

1. Begin at the root and, at each node, choose either the left or right child randomly with probability $\frac{n_L}{n_L+n_R}$ or $\frac{n_R}{n_L+n_R}$, respectively. Descend the tree to a leaf making left-right decisions in this way.
2. Upon reaching a leaf, generate the point **y** at random (uniformly) from the subspace defined by that leaf.
3. Compare **x** and **y** and make an accept/reject decision on **y**. This decision is discussed in more detail in the next subsection. If **y** is accepted, replace **x** by **y** as the current sample; else, if **y** is rejected, **x** remains the current sample.
4. If **y** was accepted, split the current leaf (containing **y**), and create two new daughter nodes, thus making more resolution available at this node if it is ever explored again.
5. Ascend the tree from the current sample to the root, updating n_L and n_R at each node.

2.2 Accept/reject decisions

The accept/reject decision of step 3 is similar to the familiar one used in simulated annealing, but must be slightly modified to compensate for the fact that **y** was not drawn from the search space with uniform probability. That is, the history of the search imposes a bias onto the search space. To see this, we consider the fact that the node values, n_L and n_R depend on two factors, the occurrence of low energy values in leaves (the history of acceptance) and simply the history of the search itself-- areas that have been searched are more likely to be searched again. We must compensate for the second phenomenon.

Step 1 of the preceding procedure for growing and searching the tree implies a $g(\boldsymbol{y}\,|\,\boldsymbol{x})$ which does not actually depend on the current $\boldsymbol{x}$. We will find this to be sufficient and henceforth will write the probability of generating a candidate $\boldsymbol{y}$ as simply $g(\boldsymbol{y})$, regardless of $\boldsymbol{x}$. Note, however, that a g of this form violates the symmetry $g(\boldsymbol{y})=g(\boldsymbol{x})$ (unless it is uniform). The resulting Metropolis procedure will *not* converge to the Gibbs distribution in general unless this asymmetry is accounted for. The following modification of Eq. (1.1) can be proven [5] to converge to the Gibbs density:

Accept **y** with probability. $\min\left(1,\frac{g(\mathbf{x})p(\mathbf{y})}{g(\mathbf{y})p(\mathbf{x})}\right)$. (2.2)

$g(\mathbf{y})$ is computed from the path of the descent down the tree by Equation

$$g(\mathbf{y})=\frac{1}{V_{\mathbf{y}}}\prod_{l}\hat{p}_l \qquad (2.3)$$

(2.3) where

$\hat{p}_l=\frac{a_l}{a_l+b_l}$, l represents the node visited at level l, a_l represents n_L or n_R, according to which direction was chosen at each l. Similarly, b_l represents the n of the direction not cho-

sen. Finally, in Eq. (2.3) we divide by V_y, the hypervolume of the leaf containing **y** to ensure that g represents a proper probability density. In [5], we show that the Gibbs distribution is a stationary state of the algorithm with the modification of Equation 2.

2.3 Annealing schedule

Following an original suggestion of Kirkpatrick et.al., [19] to cool more slowly when the specific heat is high, we use $T \leftarrow rT$, where $r = 1 - \frac{dS}{C_v}$, and dS is a small positive constant. C_v is a term easily related to the variance of the energy. This form for r is one of a family of similar schedules which have been reported in the literature[1], and differs only in the power to which C_v is raised

3 Performance

In this section, we provide examples of application of this new method for fitting data.

3.1 An asymptotic exponential

Figure 3 illustrates the result of using tree annealing to find the parameters of an exponential fit by using tree annealing.

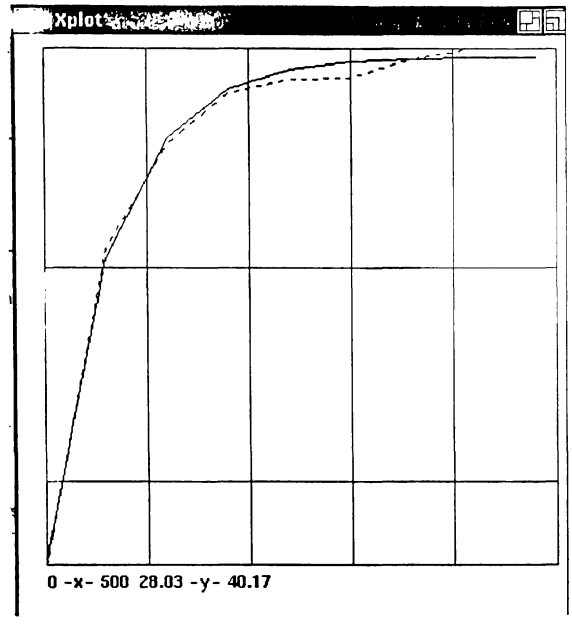

Figure 3
Results of fitting a damped exponential. Data is the dotted line , Fit is the solid line.

Our experience in attempting to find the parameters of the fit shown in Figure 3 are interesting. We repeat equation (1.1) here for reference:

$$\sum_i [(y_i - (a + be^{cx_i})]^2$$

This expression represents the squared error resulting from fitting a data set with an exponential which saturates at a non-zero asymptote. Straightforward attempts to produce a linear system of equations by taking logarithms fail, due to the additive constant a; and the minimization is therefore unsolvable analytically.

Applying simple gradient descent failed miserably, since the step size required for c is radically different from the appropriate step for a and b.

Use of second derivatives to determine the correct step size required a fair amount of tedious algebra, but did re-

sult in a descent algorithm which would work if started in the proper place. We were eventually able to determine that the error function was convex (had only one minimum), but suffered from the plateau problem discussed in section 1.1, so that if we started the descent at a large (<-.5) negative value of c, our algorithm never moved, since all derivatives were essentially zero.

This process required about two days of effort. We then typed the problem description into *INTEROPT,* (described in the next section) and TA found the solution within about 30 seconds.

3.2 Response curves

Atkinson[2] has described the usefulness of equations of the general

form $$V = r(P^{\alpha} - P_0^{\alpha})10^{-cp} \qquad (3.1)$$

,where r and α are parameters which must often be determined by data fitting.

The expression presented [2, Eq. 17] relating subjective magnitude of taste to concentration is of this general form, but with the offset term set to zero:

$$S = rw^{\alpha}\, 10^{-cw} \qquad (3.2)$$

,

where α is normally $\cong$ 1.33. Although means-squared error fits of equations of the form of Eq. 3.1 are not generally solvable by linear techniques, the more specific from of Eq. 3.2

is solvable using linear methods, simply by taking the $\log_{10}$ of both sides. However, we did not use this strategy. Instead, simply typed in the data provided in Atkinson's paper, and the expression of Eq. 3.2, and again found the optimum immediately. Interestingly, we found a better fit at α = 1.6 than at the published 1.33. Then, as an added interesting experiment, we re-introduced the offset term, putting Eq. 3.2 in the general form of Eq. 3.1,

$$S = r(w^{\alpha} + w_0^{\alpha})10^{-cW} \qquad (3.3)$$

and estimated all four parameters.

Figure 4 shows the results (compare with Fig 8 in [2], except that we plot S vs. W linearly rather than log-log.)

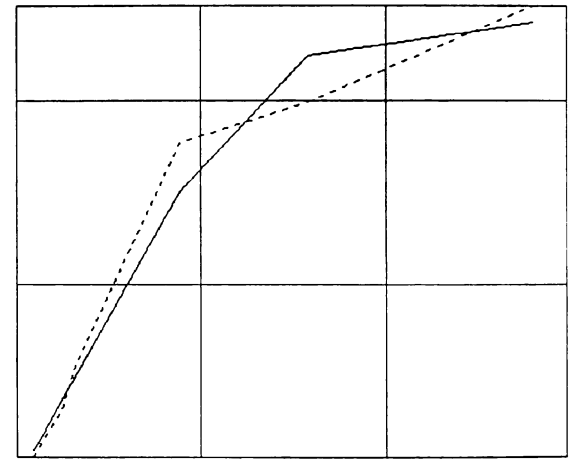

Figure 4.
Data from [2] and result of fitting w, c, and α using TA.

.Interestingly, by abandoning the need for linear methods, and allowing the W_0 term, we find that the optimal estimate for a is now 1.39, much closer to the predicted 1.33, but that W_0 is a (non-negligible) –2.76. Our purpose here is not only to show the ease with which such experiments may be performed with TA.

4 Interactive Optimization

As we used TA, and as our colleagues began to use it to solve their problems, it rapidly became apparent that a user-friendly interface to the optimization algorithm was needed. Consequently, one of the authors of this paper wrote a program called *INTEROPT*, which provides this interface. The user is asked to define his/her problem in a convenient, natural-language manner. *INTEROPT* then takes the problem definition, *writes a program*, compiles that program, and performs the optimization. In most instances, *INTEROPT* is able to solve general-purpose problems, without needing any manual intervention (such as parameter adjustments) at all. Most of the problems described in the previous section, diverse as they are, were solved with the **same software**. An example *INTEROPT* dialogue is presented in Figure 5.

5 Conclusion

In this paper, we have introduced a new technique for solving nonlinear optimization problems-- in particular, those problems which have multiple minima: Tree Annealing. TA cannot (at this time) handle more than about 30 variables. Even with this restriction, TA seems to be a very exciting method, since *NO* analysis needs to be done to solve a large class of problems. With *INTEROPT* the user simply types in the function to be minimized, and the domain of the variables, and the program does the rest.

6 References

1. E.H.L. Aarts and P.J.M. van Laarhoven. *Simulated Annealing: Theory and Applications* D. Reidel Publishing Company, Dordrecht, Holland, 1987.
2. Atkinson, William H., 1982, "A General Equation for Sensory Magnitude",*Perception and Psychophysics*, vol. 31(1), pp. 26-40.
3. V. Badami, N. Corby, N. Irwin, and K. Overton, "Using range data for inspecting printed wiring boards". In *Proc. IEEE Int. Conf. on Robotics and Automation,* Philadelphia, April, 1988.
4. H. Baker and T. Binford "A system for automated stereo mapping". In *Proc. Image Understanding Workshop* pp. 215-222 Sciences Applications, Inc. 1982.
5. G. Bilbro "A Metropolis Procedure with Asymmetric Neighborhoods" Technical report TR- Center for Communications and Signal Processing, North Carolina State University, Raleigh, 1990.
6. G. Bilbro, M. Steer, R. Trew, and S. Skaggs. "Parameter extraction of microwave transistors using tree annealing", IEEE/Cornell Conf. Digest, August, 1989.
7. C. Chow and T. Kanako "Automatic boundary detection of the left ventricle from cineangiograms" *Computers in Biomedical Research*, **5**:388-409, 1972.
8. G. Dahlquist and A. Bjorck *Numerical Methods*, Prentice-Hall, Englewood Cliffs, NJ 1974.
9. R. Duda and D. Nitzan. "Low level processing of registered intensity and range data." In *Proc. 3rd In.t Joint Conf. Artificial Intelligence,* 1976.

10.O. Faugeras et. al. "Towards a flexible vision system", In *Robot Vision* A. Pugh, ed., pp. 129-142. IFS, UK 1982.

11.O. Faugeras and M. Herbert "The representation, recognition, and locating of 3-D objects", *International Journal of Robotics Research,* **5**(3), Fall, 1986.

12.Greenlee, Mark W. and Magnussen, Svein, 1987, "Saturation of the Tilt Aftereffect", *Vision Research*, vol. 27, no. 6, pp. 1041-1043.

13.W. Grimson *From Images to Surfaces: A Computational Study of the Human Early Vision System".* MIT Press, Cambridge, 1982.

14.Y. Han, W. Snyder, and G. Bilbro, "Determination of optimal pose using tree annealing", In *Proc. IEEE Int. Conf. on Robotics and Automation,* Cincinnati, May, 1990.

15.M. Hollins, A. Goble, B. Whitsel, and M. Tommerdahl, "Time course and action spectrum of vibrotactile adaptation", *Somatosensory and Motor Research*, In review

16.R. Jarvis "A mobile robot for computer vision research". In *Proc. Third Australian Comp. Sci. Conf., Australian Nat. Univ. Canberra, ACT, 1980.*

17.R. Jarvis, "A laser time-of-flight range scanner for robotic vision", TR TR-CS-81-10, Comp. Sci. dept., Australian Nat. Univ. Canberra, 1981.

18.Kalos, M.H. and P. A. Whitlock, *Monte Carlo Methods, Volume 1: Basics*, Wiley and Sons, New York, 1986.

19.S. Kirkpatrick, C. Gelatt, and M. Vecchi, "Optimization by simulated annealing", *Science*, 220(4598):671-680, May 1983.

20.R. Lewis and A. Johnston, "A scanning laser rangefinder for a robotic vehicle. In *Proc 5th Int. Joint Conf Artificial Intelligence* 1977.

21.Magnussen, Svein and Johnson, Tore, "Temporal Aspects of Spatial Adaptation: A Study of the Tilt Aftereffect", *Vision Research,* vol. 26, no. 4, pp. 661-672.1986.

22.D. Marr and T. Poggio. "A computational theory of human stereo vision",. In *Proc. Royal Society of London.* (204), 1979.

23.Metropolis, N., A. Rosenbluth, M. Rosenbluth, A. Teller and E. Teller, "Equation of State Calculations by Fast Computing Machines," *J. of Chem. Physics*, 21 (1953) 1087-1092.}

24.U. Mishra, A. Brown, and S. Rosenbaum, "DC and RF performance of 0.1υ gate length $Al_{.48}In_{.52}As$ - $Ga_{.38}In_{.62}As$ pseudomorphic hemts" *IEDM Technical Digest* December, 1988.

25.H. Nishihara "Prism: A practical real-time imaging stereo system". In *Proc. of 3rd Int. conf. on Robot Vision and Sensory Control.* R. Brooks, ed. North-Holland, Amsterdam. 1983.

26.D. Nitzan, A. Brain, and R. Duda,"The measurement and use of registered reflectance and range data in scene analysis." *Poc. IEEE* **65**,:206-220, Feb. 1977.

27.Y. Ohta and T. Kanade, "Stereo by intra- and interscanline search using dynamic programming" TR CMU-CS-83-162, Carnegie Mellon U. Pittsburgh, 1983.

28.M. Oshima and Y. Shirai "Object recognition using 3-D information", *IEEE PAMI*, **PAMI-5**(4):353-361, 1983.

29.R. Paul. *Robotic Manipulators, Mathematics, Programming and Control* MIT Press, Cambridge, Mass. 1981.

30.M. Peleg and O. Campanella, "On the Mathematical form of psychophysical relationships with special focus on the perception of mechanical properties of solid objects", *Perception and Psychophysics*, vol 44, no 5, 1988.

31. R. Scobey and C. Johnson, "Psyophysical properties of displacement thresholds for moving targets", *Acta Psychologica* vol 48, 1981

32.W. Snyder *Industrial Robots, Computer Interfacing and Control"* Prentice-Hall, 1984.

33.W. Snyder, G. Bilbro, A. Logenthiran, and S. Rajala "Optimal Thresholding-- a New Approach" *Pattern Recognition Letters* In review.

34.B. Spain *Analytic Quadrics*. Pergamon Press, 1960.

35.Y. Tsuchiya, E. Inuzuka, Y. Suzui, and W. Yu. "Ultrafast streak camera", In *Proc. 13th Int. Congr. High Speed Photograph and Photonics"*, Tokyo, Japan, August 1978.

36. D. Vanderbilt and G. Louie, "A Monte Carlo Simulated Annealing Approach to Optimization over Continuous Variables", *J Comput. Phys.* **36**(1984)259-271.

37.Virsu, Veijo and Lee, Barry B., "Light Adaptation in Cells of Macaque Lateral Geniculate Nucleus and Its Relation to Human Light Adaptation", *Journal of Neurophysiology*, vol. 50, no. 4. 1983

```
%interopt
Do you want to optimize a function?no
Well then, would you like to fit some data?yes
How many parameters does the fit have?3
How many variables does the fit have?1
How many outputs does the fit have?1
Data file name?demos/data1
How many data points are in that file?9
Now I need to know the parameters over which I should minimize
What is the name of parameter number 0?a
What is the name of parameter number 1?b
What is the name of parameter number 2?c

Now, I need to know the names of the variables

DONT USE y!
What is the name of variable number 0
x
Enter the function to be fit
use the form  y[i] = f(x), with a different y[i] for each output
y[0] = a + b * exp ( c * x)
I'm now compiling your function. Please wait
I'm now compiling the main program. Please wait
OK. We got that far. Now, tell me the max and min values on each variable
What is the lower limit of a?20
What is the upper limit of a?50
What is the hat is the lower limit of b?-20
What is the upper limit of b?20
hat is the lower limit of c?-1
What is the upper limit of c?0
I am now running your optimization:
```

Figure 5
Example application of *INTEROPT*. A dialogue in which the user defines a problem of fitting the sum of two sine waves.