Appendix A

THE ENVIRONMENT E

A.1 Introduction

In order to evaluate the capabilities of different adaptive strategies, an environment E was chosen to include instances of performance measures representing a broad class of functions. For convenience, each function in E was defined to be a performance index to be minimized. Care was taken to include instances of continuous, discontinuous, convex, non-convex, unimodal, multimodal, quadratic, non-quadratic, low-dimensional and high-dimensional functions as well as functions with Gaussian noise.

For testing purposes, each function was restricted to a bounded subspace of R^n of the form $a_i \leq x_i \leq b_i$, $i=1,\ldots,n$. Within this subspace each function was discretized by specifying a resolution factor Δx_i for each axis. Since the genetic algorithms use a binary representation of the search space, the number of discrete points on each axis, $(b_i-a_i)/\Delta x_i + 1$, was chosen to be a power of 2 so that a direct comparison with alternative adaptive plans could be made.

A.2 Test Function F1

Test function F1 is given by:

$$F1(X) = \sum_{i=1}^{3} x_i^2$$

F1 is a simple 3-dimensional parabola with spherical constant-cost contours. It is a continuous, convex, unimodal, low-dimensional quadratic function with a minimum of zero at the origin. Because of its simplicity and symmetry, F1 provides an easily analyzable first test for an adaptive plan. For testing purposes F1 was restricted to the space A defined by $-5.12 \leq x_i \leq 5.12$, $i=1,2,3$ with a resolution factor $\Delta x_i = .01$ on each axis. So the space A to be searched consisted of $(1024)^3 \cong 10^9$ alternative solutions on which:

$$\mathrm{MAX}(F1) = F(\pm 5.12, \pm 5.12, \pm 5.12) = 78.6$$

$$\mathrm{MIN}(F1) = F(0,0,0) = 0$$

$$\mathrm{AVE}(F1) = \frac{1}{(10.24)^3} \int_A F1(X)\, dX = 26.2$$

Figures A.1a and A.1b illustrate the surface defined by F1 in its two-dimensional form.

A.3 <u>Test Function F2</u>

Test function F2 is given by:

$$F2(X) = 100*(x_1^2 - x_2)^2 + (1-x_1)^2$$

F2 is a standard test function in the optimization literature, first proposed by Rosenbrock(1960). It is a continuous, non-convex, unimodal, low-dimensional quartic function with a minimum of zero at (1,1). It is a difficult minimization problem because it has a deep

FIG. A.1A: 2-DIMENSIONAL VERSION OF F1

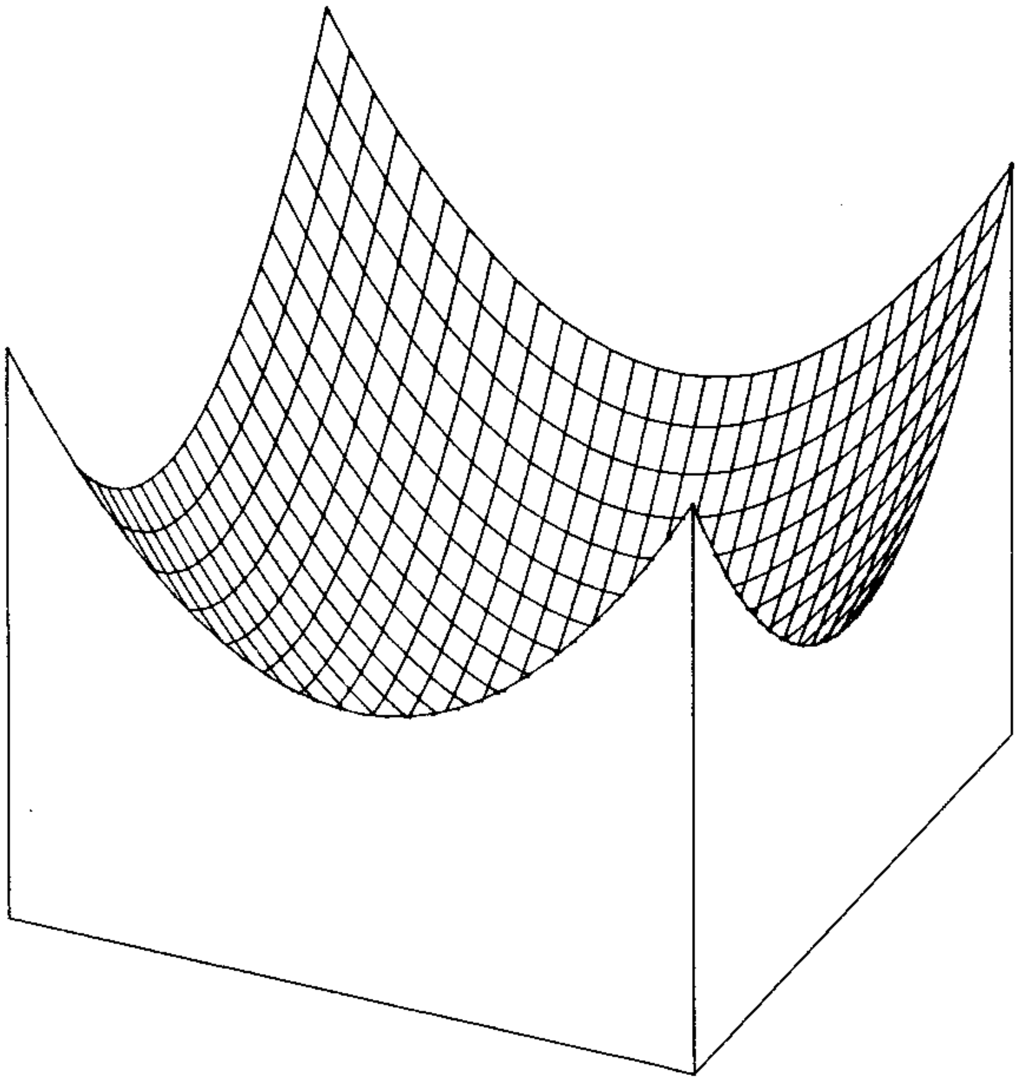


Figure A.1a: Top surface defined by the 2-dimensional version of F1.

FIG. A.1B: INVERTED 2-DIMENSIONAL VERSION OF F1

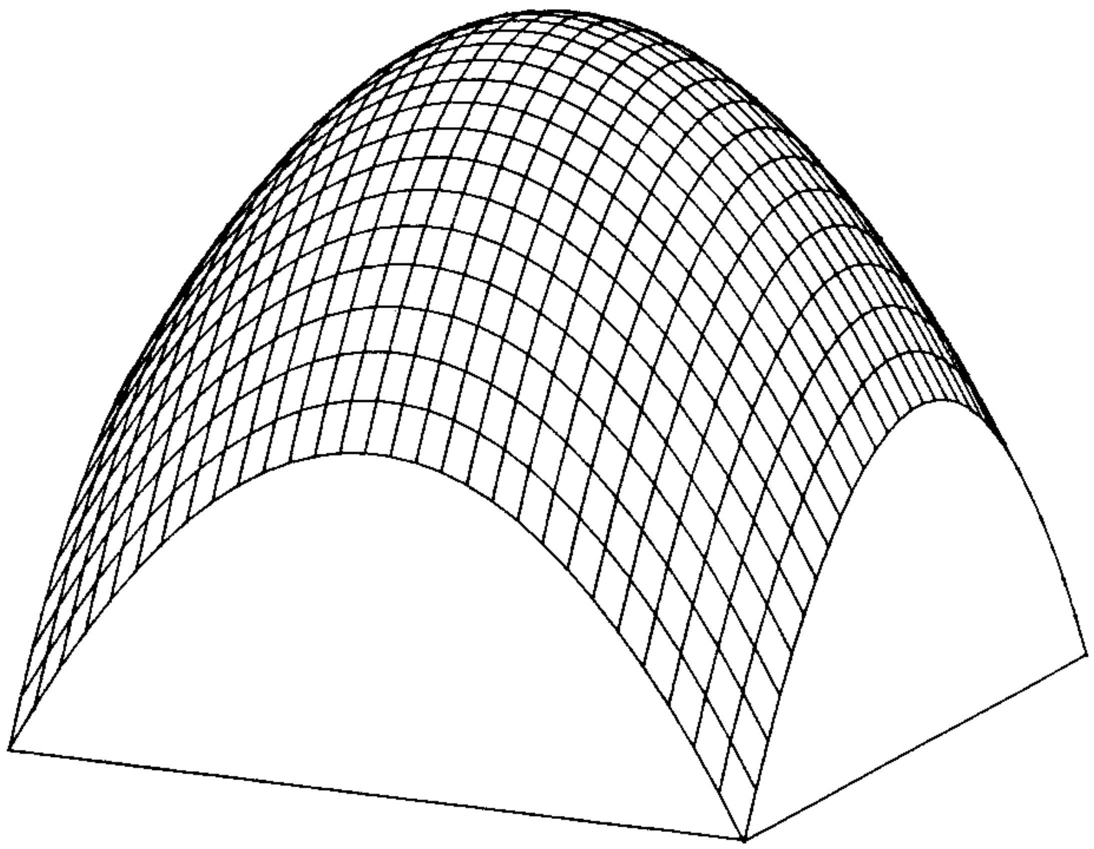


Figure A.1b: Bottom surface defined by the 2-dimensional version of F1.

parabolic valley along the curve $x_2 = x_1^2$. For testing purposes, F2 was restricted to the space A defined by $-2.048 \leq x_i \leq 2.048$, i=1,2 with a resolution factor of $\Delta x_i = .001$ along each axis. So the space A to be searched consisted of $(4096)^2 \cong 1.7*10^6$ alternative solutions on which:

$$\text{MAX}(F2) = F(-2.048,-2.048) = 3905.93$$

$$\text{MIN}(F2) = F(1,1) = 0$$

$$\text{AVE}(F2) = \frac{1}{(4.096)^2} \int_A F2(X)\, dX = 494.05$$

Figures A.2a and A.2b illustrate the surface defined by F2.

A.4 Test Function F3

Test function F3 is given by:

$$F3(X) = \sum_{i=1}^{5} [x_i]$$

where $[x_i]$ represents the greatest integer less than or equal to x_i. Hence, F3 is a 5-dimensional step function. It is a discontinuous, non-convex, unimodal function of moderate dimension which is piece-wise constant. F3 was chosen as a test for handling discontinuities. For testing purposes, F3 was restricted to the space A defined by $-5.12 \leq x_i \leq 5.12$ with a resolution factor of $\Delta x_i = .01$ on each axis. So the space A to be searched consisted of $(1024)^5 \cong 10^{15}$ alternative solutions on which:

FIG. A.2A: F2 (ROSENBROCK'S FUNCTION)

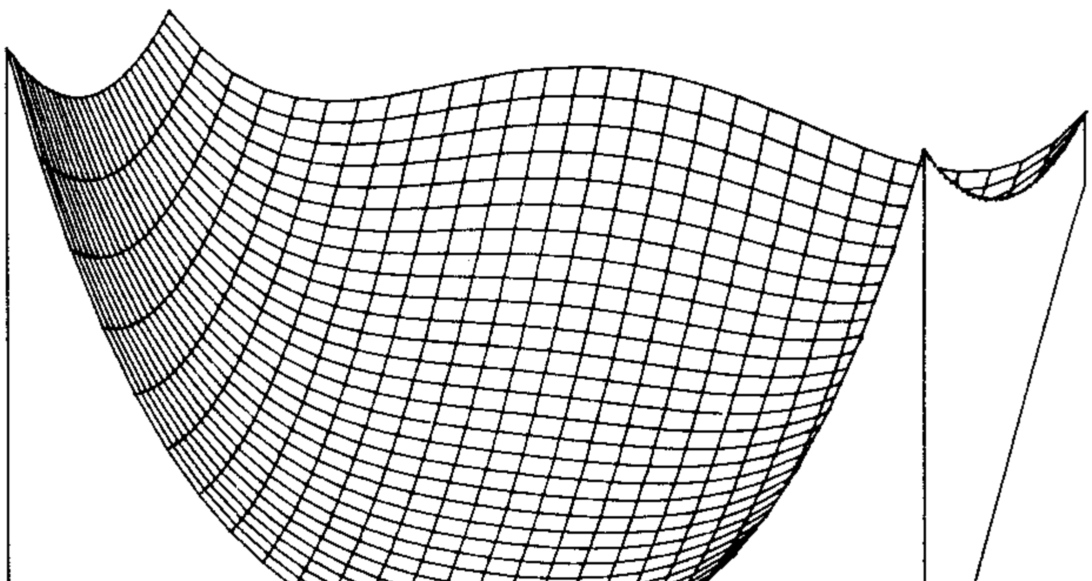


Figure A.2a: Top surface defined by test function F2.

FIG. A.2B: INVERTED F2 (ROSENBROCK'S FUNCTION)

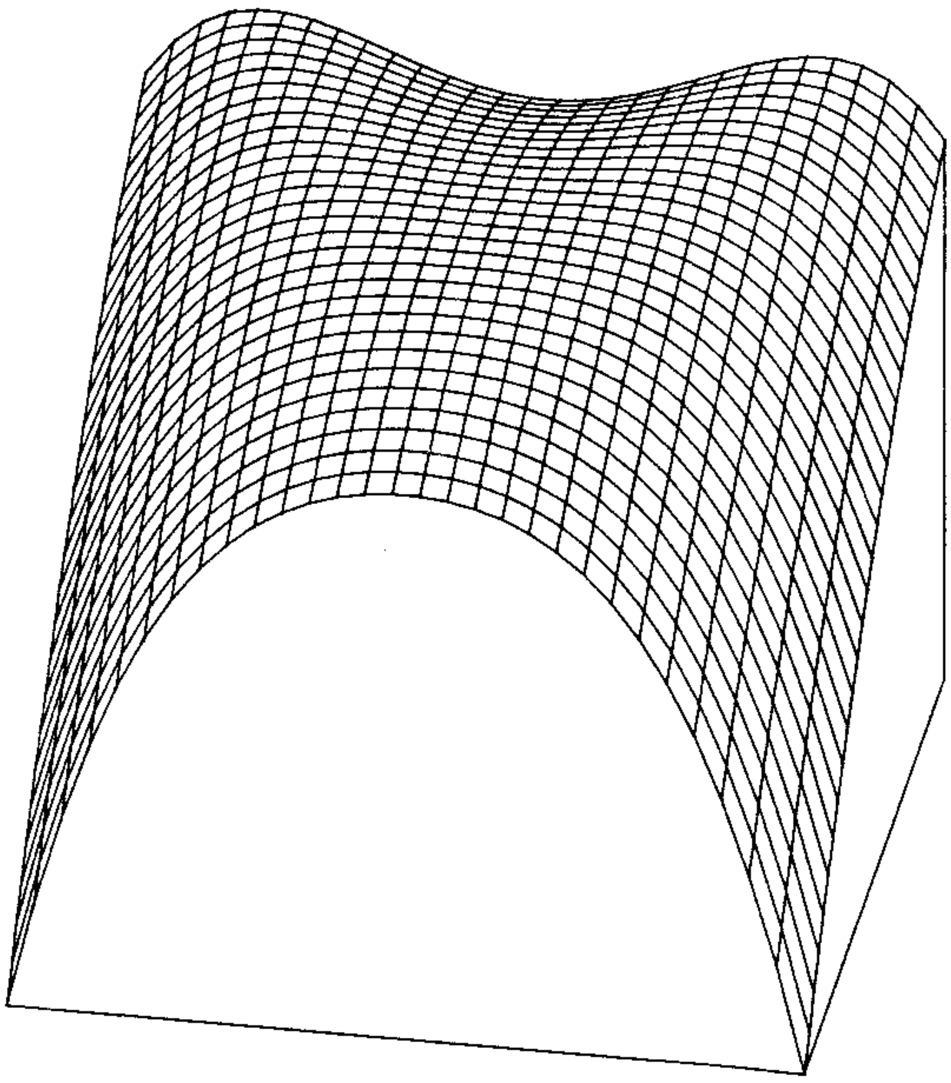


Figure A.2b: Bottom surface defined by test function F2.

$$MAX(F3) = F3(5.12,5.12,5.12,5.12,5.12) = 25$$

$$MIN(F3) = F3(-5.12,-5.12,-5.12,-5.12,-5.12) = -30$$

$$AVE(F3) = \sum_{i=1}^{5} AVE\left[x_i\right] = -2.5$$

Figure A.3 illustrates the surface defined by F3 in its two-dimensional form.

A.5 Test Function F4

Test function F4 is given by:

$$F4(X) = \sum_{i=1}^{30} ix_i^4 + GAUSS(0,1)$$

F4 is a continuous, convex, unimodal, high-dimensional quartic function with Gaussian noise. For testing purposes, F4 was restricted to the space A defined by $-1.28 \leq x_i \leq 1.28$, i=1,...,30 with a resolution factor of $\Delta x_i = .01$ on each axis. So the space A to be searched consisted of $(256)^{30} \cong 10^{72}$ alternative solutions on which:

$$MAX(F4) = F4(\pm 1.28, \pm 1.28, \ldots, \pm 1.28) = 1248.2$$

$$MIN(F4) = F4(0,0,\ldots,0) = 0$$

$$AVE(F4) = 249.6$$

Figures A.4a and A.4b illustrate the surface defined by F4 in its two-dimensional form without Gaussian noise.

A.6 Test Function F5

Test function F5 is given by:

FIG. A.3: 2-DIMENSIONAL VERSION OF F3

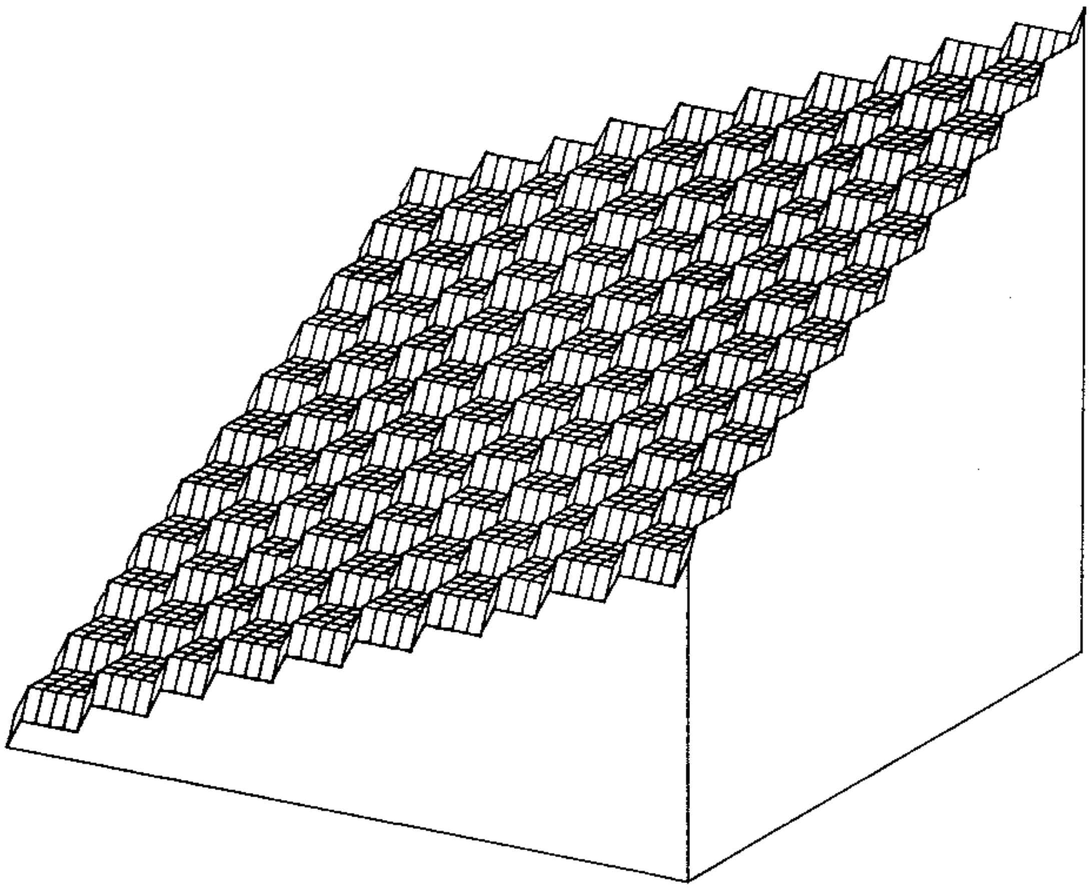


Figure A.3: Surface defined by the 2-dimensional version of test function F3.

FIG. A.4A: 2-DIMENSIONAL VERSION OF F4

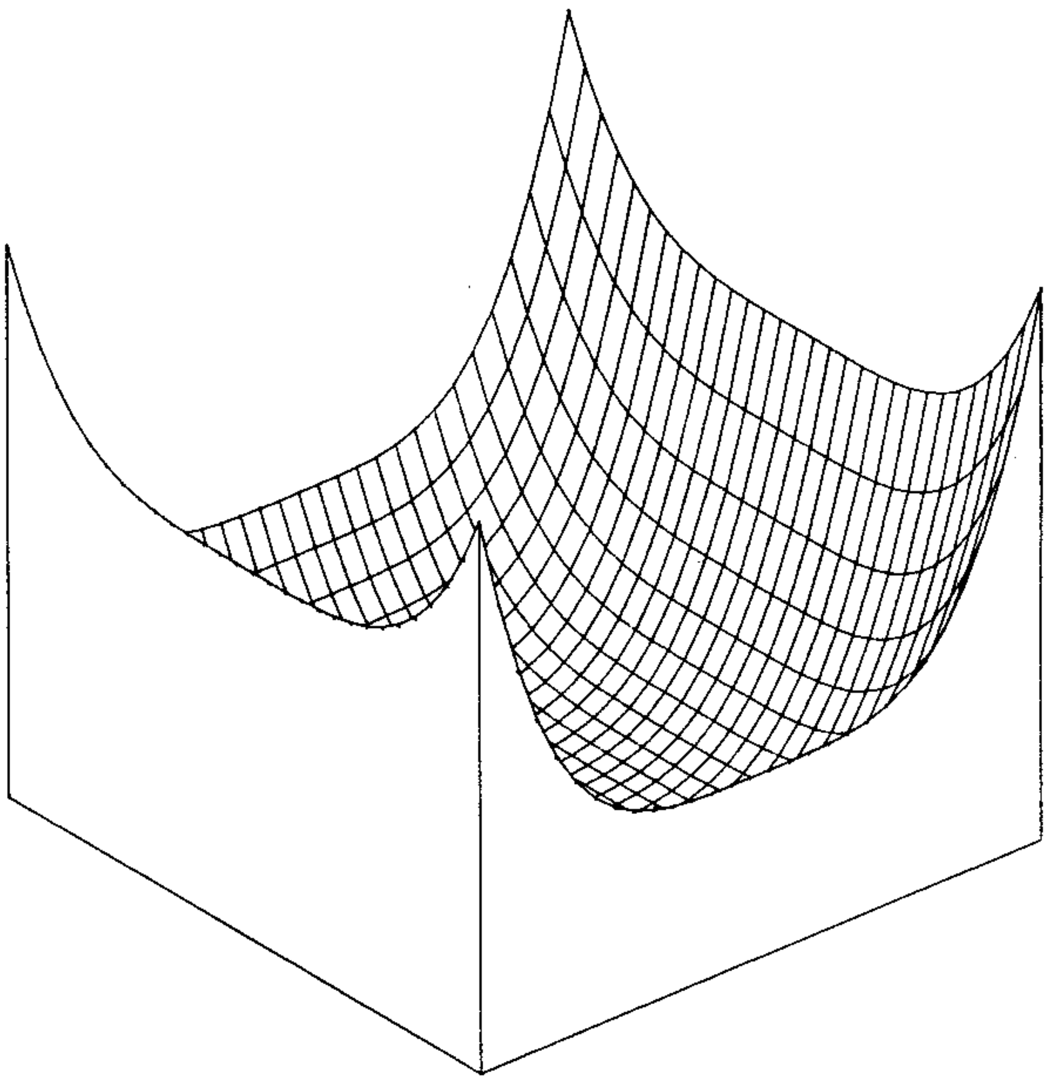


Figure A.4a: Top surface defined by the 2-dimensional version of test function F4.

FIG. A.4B: INVERTED 2-DIMENSIONAL VERSION OF F4

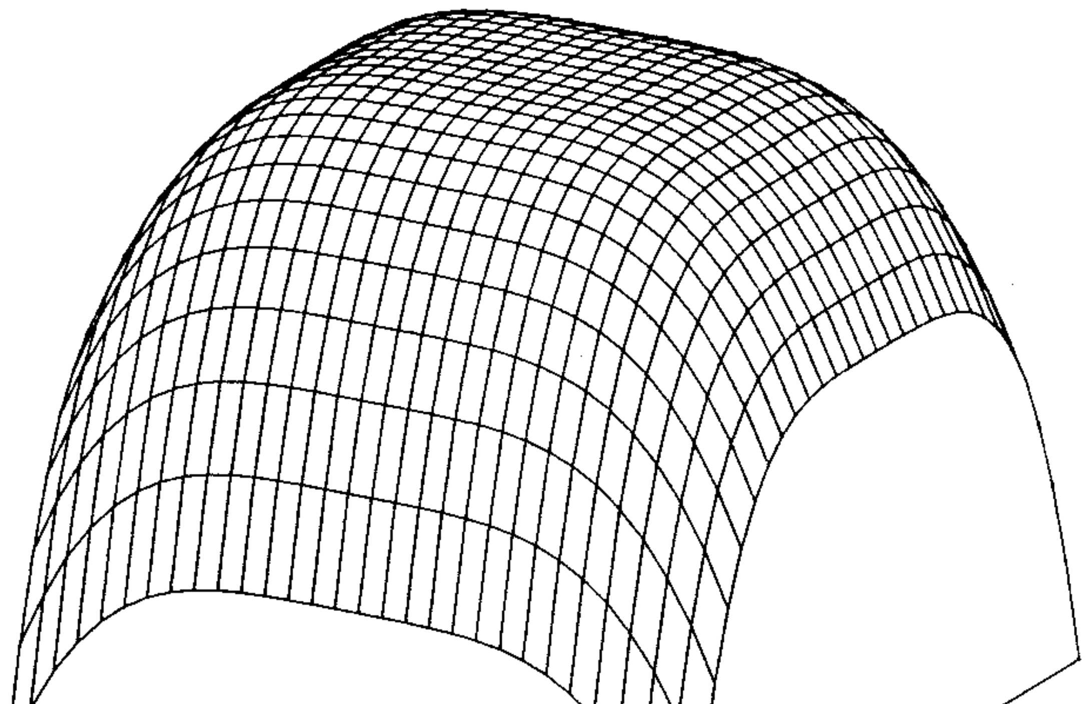


Figure A.4b: Bottom surface defined by the 2-dimensional version of test function F4.

$$\frac{1}{F5(X)} = \frac{1}{K} + \sum_{j=1}^{25} \frac{1}{f_j(X)}$$

where

$$f_j(X) = c_j + \sum_{i=1}^{2} (x_i - a_{ij})^6$$

F5 is an interesting multimodal function synthesized as suggested by Shekel (1971). It is a continuous, non-convex, non-quadratic, two-dimensional function with 25 local minima approximately at the points $\left\{(a_{1j},a_{2j})\right\}_{j=1}^{25}$. The function value at the point (a_{1j},a_{2j}) is approximately c_j.

For testing purposes, the a_{ij} were defined by:

$$\left[a_{ij}\right] = \begin{bmatrix} -32,-16,\ 0,\ 16,\ 32,-32,-16,\ldots,\ 0,16,32 \\ -32,-32,-32,-32,-32,-16,-16,\ldots,32,32,32 \end{bmatrix}$$

with $c_j = j$ and $K = 500$. F5 was restricted to the space A defined by $-65.536 \leq x_i \leq 65.536$, i=1,2 with a resolution factor of $\Delta x_i = .001$ on each axis. So the space A to be searched consisted of $(131{,}072)^2 \cong 16*10^9$ alternative solutions on which:

$$\text{MAX}(F5) \cong 500$$

$$\text{MIN}(F5) \cong 1$$

$$\text{AVE}(F5) \cong 473$$

Figures A.5a and A.5b illustrate the surface defined by F5. It is essentially a flat surface F5(X) = 500 with 25 deep perforations centered about the points (a_{1j},a_{2j}).

Near the point (a_{1j}, a_{2j}), F5 is almost completely dominated by the term $f_j(X)$, i.e.

$$F5(X) = c_j + \sum_{i=1}^{2} (x_i - a_{ij})^6$$

and hence $F5(a_{1j}, a_{2j}) \cong c_j \overset{d}{=} j$. So F5 has 25 local minima at which F5 takes on the values 1,2,....,25.

FIG. A.5A: F5 (25 FOX HOLES)

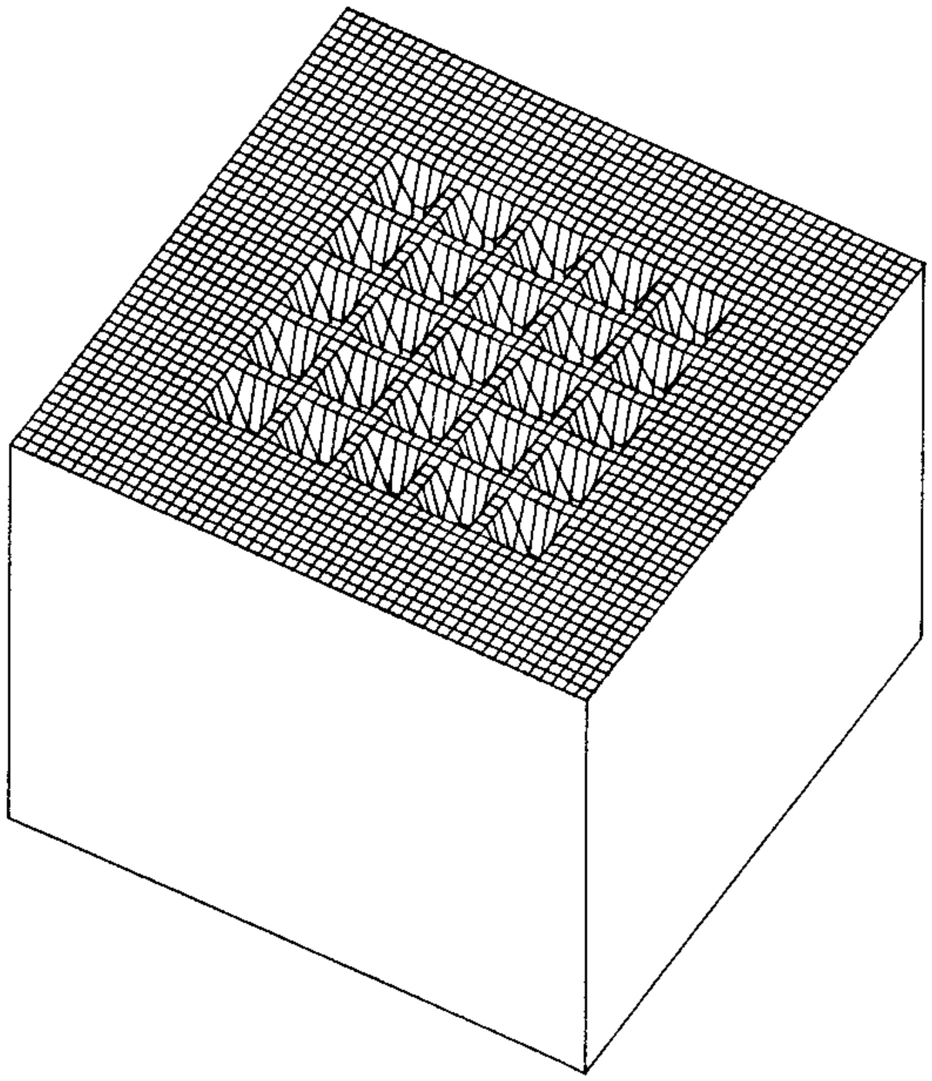


Figure A.5a: Top surface defined by test function F5.

FIG. A.5B: INVERTED F5 (25 FOX HOLES)

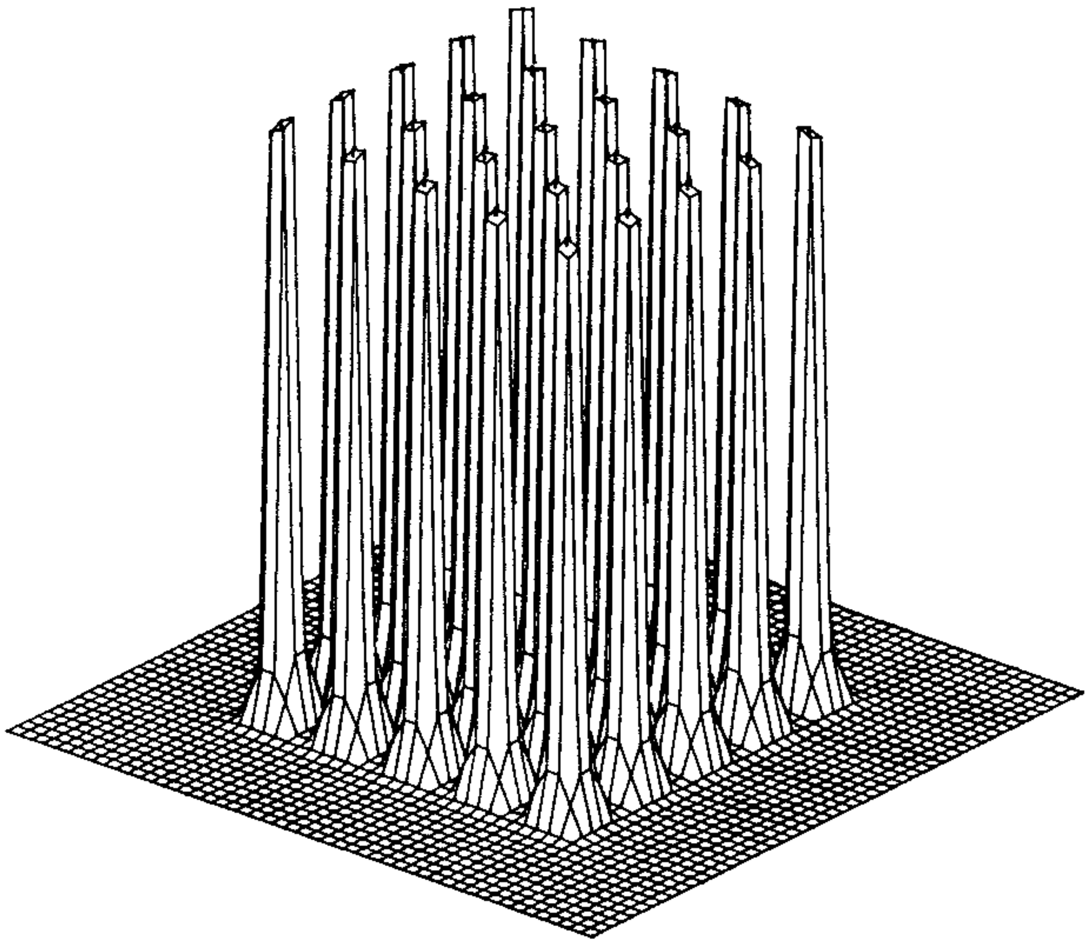


Figure A.5b: Bottom surface defined by test function F5.

Appendix B

RANDOM SEARCH ON E

B.1 Introduction

As a first attempt at evaluating genetic adaptive plans, their performance was compared with pure random search on the environment described in appendix A. Since pure random search takes advantage of none of the information accumulating over a sequence of trials, its performance serves as a lower bound on the performance of an adaptive plan. Any plan which claims to dynamically exploit sampling information had better show improved performance over random search.

In order to evaluate the performance of pure random search on the environment E, a plan RANDOM was implemented in PL/I to simulate a uniformly-distributed random search over the discrete spaces A associated with the test functions. Considerable difficulty was encountered in validating the uniform randomness of RANDOM on the multi-dimensional spaces associated with the test functions. Initially, RANDOM called the standard system library random number generator URAND N times to produce a point in N-space. However, this resulted in a non-uniform distribution biased toward a hypersphere in the center of the space. This approach was modified to maintain N separate pseudo-random streams in parallel, one for each axis. This improved the randomness in N-space somewhat, although considerable difficulties arose in determining a method for selecting N seeds so that the streams were in fact independent.

At this point I began reading about random number generators and discovered Marsaglia's article (1968) pointing out the difficulties in generating uniform distributions in N-space with multiplicative congruential random number generators, of which URAND was an instance. This led me to papers by Tootill, etc. (1971, 1973) describing the properties of several classes of Tausworthe generators, first suggested by Tausworthe (1965), which are based on primitive polynomials over GF(2) and are implemented by linear shift register sequences. These generators can be shown analytically to generate uniformly random distributions in N-space when $N \leq K$. The value of K depends on the particular member of the class chosen, and the cost of generating the random numbers increases with K. A member of the class for which K=30 was chosen and implemented as the subroutine TRAND for use with RANDOM. This resulted in a considerable improvement in the uniform randomness of RANDOM and led to very little difficulty in validating the performance of RANDOM on the environment E.

B.2 Validating RANDOM on E

In order to provide a comparison with genetic algorithms, both the off-line and on-line performance of RANDOM was measured for each of the test functions. Care was taken to attempt to validate the performance curves obtained from the simulated random searches. That is, I attempted to verify that performance curves approximated the expected performance and did not in fact represent an artifact of the pseudo-uniform distribution over the space. The verification took two forms. Whenever possible, I derived an analytic expression for the expected performance curve assuming a uniform distribution. If this was not possible, the space was rotated and inverted during the sequence of simulations used to generate a performance curve in an attempt to counteract any pseudo-random bias.

Expected off-line performance was derived as follows. If we consider a test function F as a random variable on the space A, we can ask: what is the expected number of trials required to find a point X in A such that $F(X) \leq \epsilon$. This is a standard waiting time problem for a sequence of Bernouli trials. That is, let p be the probability of a success ($F(X) \leq \epsilon$). Then the expected number of trials required to generate the first success is simply $f = \frac{1}{p}$. So the problem reduces to one of computing PROB $\left[F(X) \leq \epsilon\right]$. Whether or not this is easily computed depends, of course, on the particular function.

For on-line performance, we need to know the expected value of a sample at time t. For uniform random search, this is simply the average value of F on the space A. Whether or not this is easily computed depends again on the particular test function.

B.3 RANDOM on F1

F1 provides a good validation test for RANDOM since it is amenable to mathematical analysis. For off-line performance, we need to compute PROB $\left[F1(X) \leq \epsilon\right]$. We note that, for F1, the points in A satisfying this constraint lie in the sphere defined by $x_1^2+x_2^2+x_3^2 = \epsilon$. Moreover, the search space A is bounded by constraints of the form $|x_i| \leq b$. Hence, for $\sqrt{\epsilon} \leq b$, we have:

$$\text{PROB}\left[F1(X) \leq \epsilon\right] = \frac{V_s(\sqrt{\epsilon}\,)}{V_c(2b)}$$

where $V_s(\sqrt{\epsilon}\,)$ represents the volume of a sphere of radius $\sqrt{\epsilon}$ and $V_c(2b)$ represents the volume of a cube with sides of length 2b. The volume of a sphere is given by:

$$V_s(\sqrt{\epsilon}\,) = \frac{4}{3}\pi\epsilon^{3/2}$$

so that

$$\text{PROB}\left[F1(X) \leq \epsilon\right] = \frac{4\pi\epsilon^{3/2}}{3(2b)^3} = \frac{\pi\epsilon^{3/2}}{6b^3}$$

and hence the expected number of trials required to locate a point satisfying the constraint is given by:

$$f = \frac{1}{\text{PROB}\left[F1(X) \leq \epsilon\right]} = \frac{6b^3}{\pi \epsilon^{3/2}}$$

Expected on-line performance is obtained by computing:

$$\text{AVE}(F1(X)) = \frac{1}{(2b)^3} \int_A F1(X)\, dX$$

$$= \frac{1}{(2b)^3} \int\int_{-b}^{b}\int x_1^2+x_2^2+x_3^2\ dx_1 dx_2 dx_3$$

$$= \frac{1}{(2b)^3} * 8b^5$$

$$= b^2$$

Figures B.1a and B.1b compare the expected performance curves with those generated by RANDOM. As can be seen, the values agree closely.

B.4 RANDOM on F2

Deriving an analytic expression for the off-line performance of RANDOM on F2 is very difficult, if not impossible. As a consequence, no direct validation of the performance curve illustrated in figure B.2a was done. However, the space A was rotated and inverted during the generation of the performance curve in an

FIG. B.1A: OFF-LINE PERFORMANCE ON F1

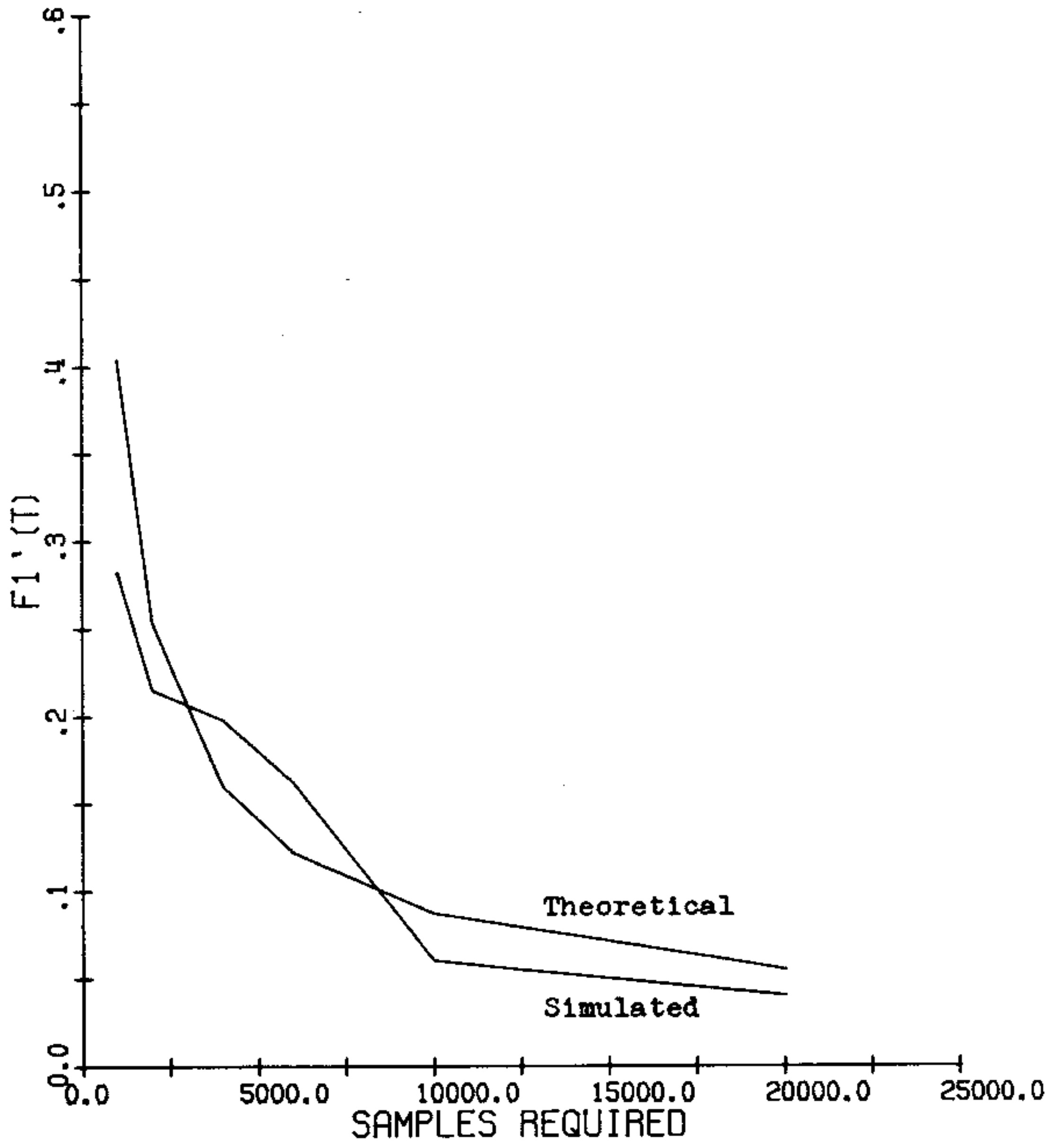


Figure B.1a: Off-line performance curves for random search on test function F1.

FIG. B.1B: ON-LINE PERFORMANCE ON F1

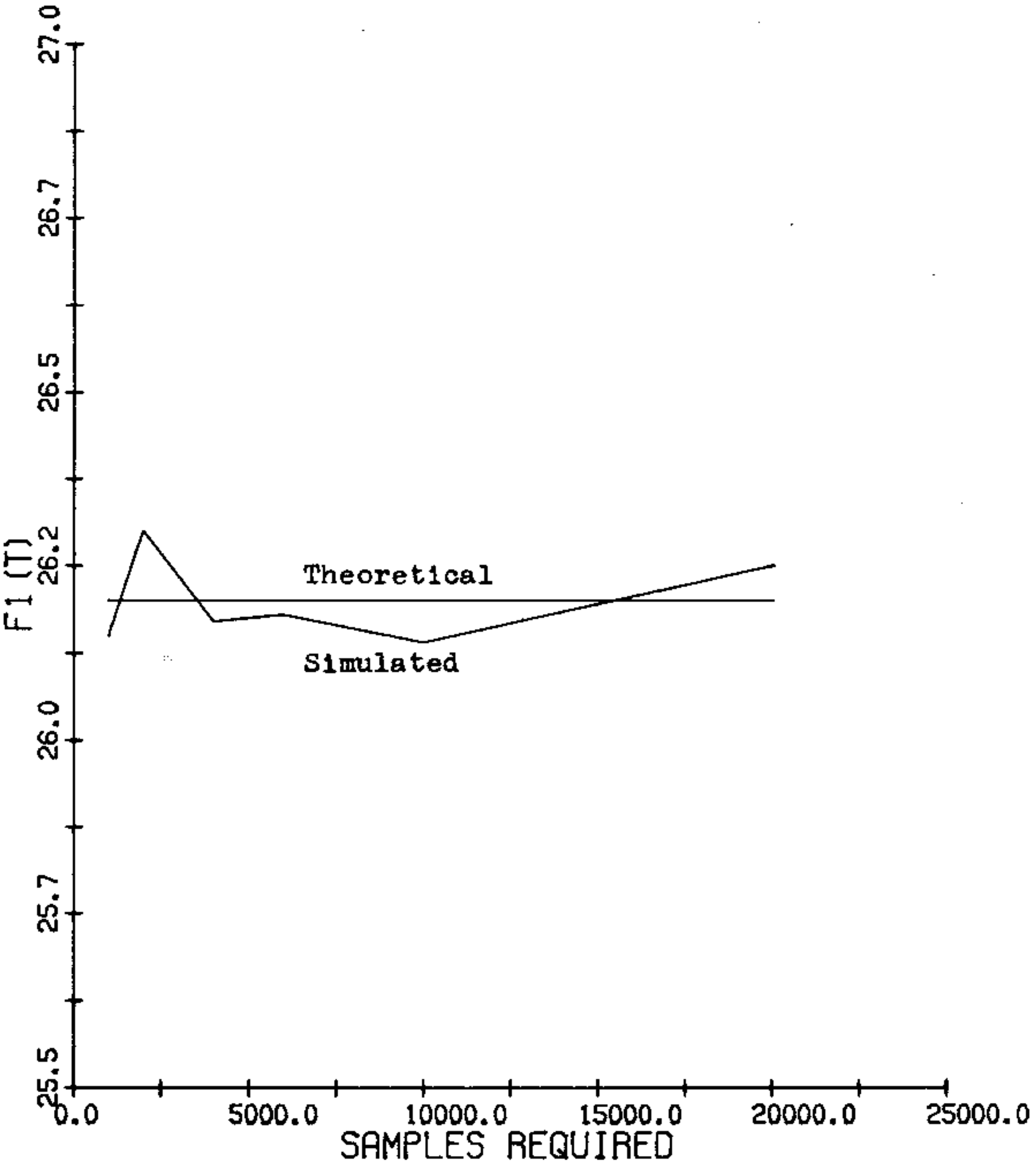


Figure B.1b: On-line performance curves for random search on test function F1.

FIG. B.2A: OFF-LINE PERFORMANCE ON F2

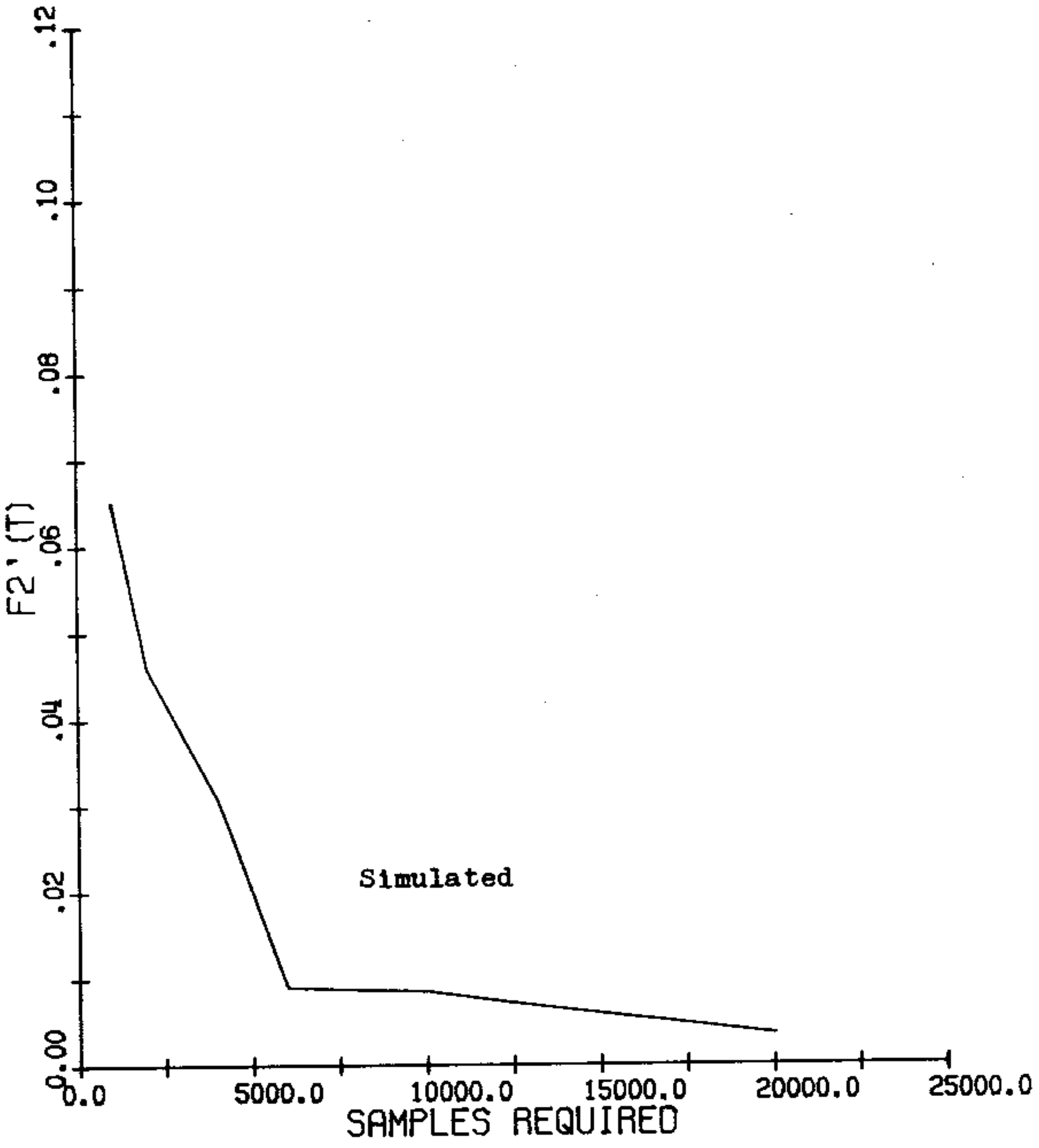


Figure B.2a: Off-line performance curve for random search on test function F2.

attempt to minimize any bias.

Expected on-line performance on F2 is computed as follows:

$$\mathrm{AVE}(F2(X)) = \frac{1}{(2b)^2} \int_A F2(X)\, dX$$

$$= \frac{1}{(2b)^2} \int_{-b}^{b}\!\!\int 100(x_1^2-x_2)^2+(1-x_1)^2\, dx_1 dx_2$$

$$= \frac{1}{4b^2}\ 80b^6+\frac{404}{3}b^4+4b^2$$

$$= 20b^4 + \frac{101}{3}\, b^2 + 1$$

Figure B.2b compares the theoretical on-line performance with that generated by RANDOM. The agreement is quite good.

B.5 RANDOM on F3

Deriving the expected off-line performance on F3 is a straightforward, but tedious, problem in combinatorics. The probability of landing on a particular plateau is given by:

$$p = \frac{1}{(2b)^5} \prod_{i=1}^{5} \ell_i$$

where ℓ_i are the lengths of the sides of the plateau. It remains only to count the number of plateaus satis-

FIG. B.2B: ON-LINE PERFORMANCE ON F2

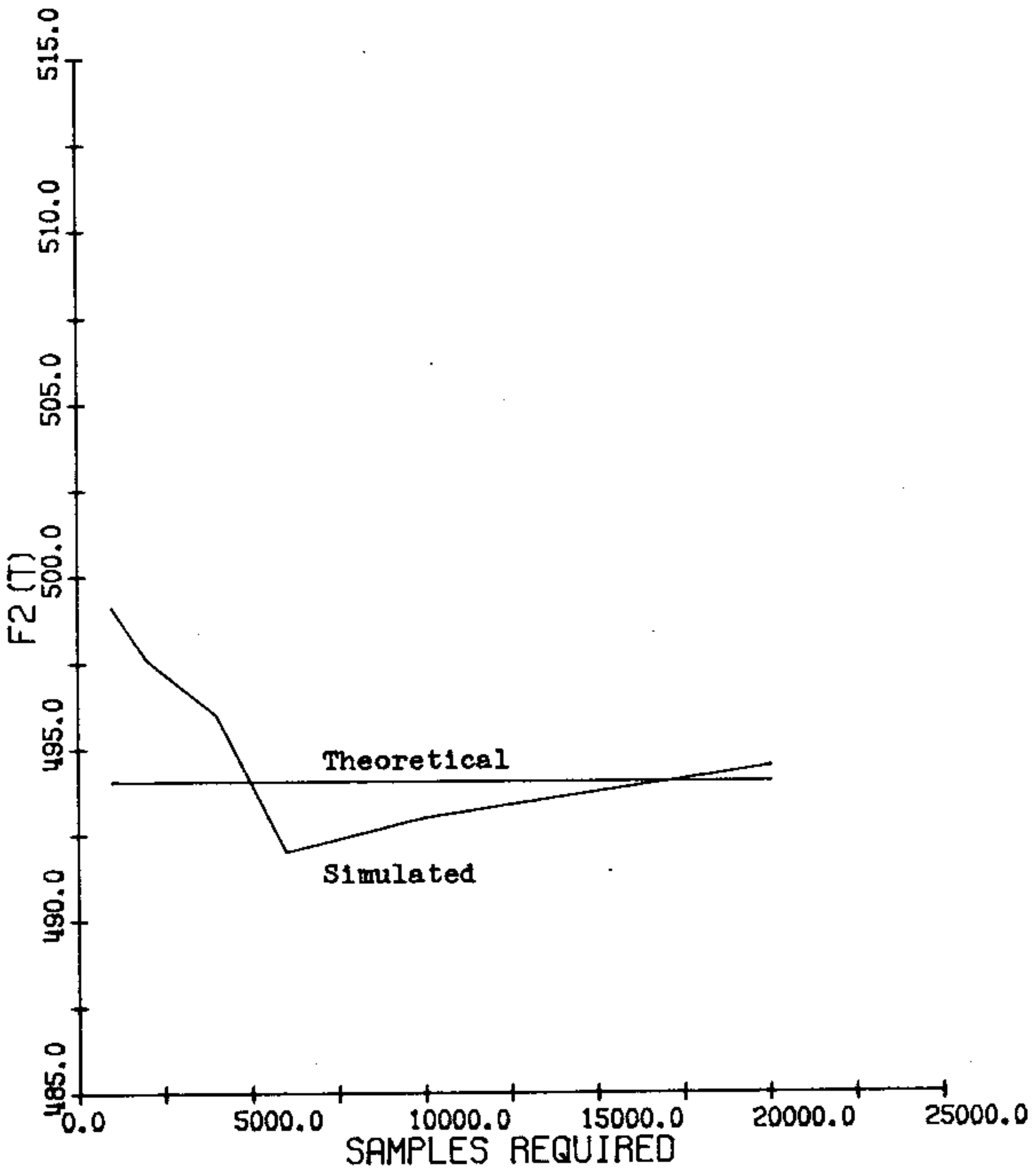


Figure B.2b: On-line performance curves for random search on test function F2.

fying the constraint $F3(X) \leq \epsilon$ and sum their associated probabilities. For example, the PROB $\left[F3(X) \leq -29\right]$ when $|x_i| \leq 5.12$ is computed as follows:

Since $F3(X) = \sum_{1}^{5} \left[x_i\right]$, the minimum value $F3(X)=-30$ can only be obtained by $\left[x_i\right] = 6$ for all i. Hence,

$$\text{PROB}\left[F3(X)=-30\right] = \left(\frac{.12}{10.24}\right)^{5}$$

Similarly, $F3(X)=-29$ is obtained by summing

$$(-6)+(-6)+(-6)+(-6)+(-5)$$

so that

$$\text{PROB}\left[F3(X)=-29\right] = \binom{5}{1}\left(\frac{.12}{10.24}\right)^{4}\left(\frac{1}{10.24}\right)$$

and summing the two yields PROB $\left[F3(X) \leq -29\right]$.

Expected on-line performance on F3 is given by:

$$\text{AVE}(F3(X)) = \text{AVE} \sum_{1}^{5} \left[x_i\right]$$

$$= \sum_{1}^{5} \text{AVE} \left[x_i\right]$$

$$= 5 * \left(\left[-b\right] + \left[b\right] \right)/2$$

Figures B.3a and B.3b compare the expected performance curves with those obtained from RANDOM. As can be seen, the values are very close.

FIG B.3A: OFF-LINE PERFORMANCE ON F3

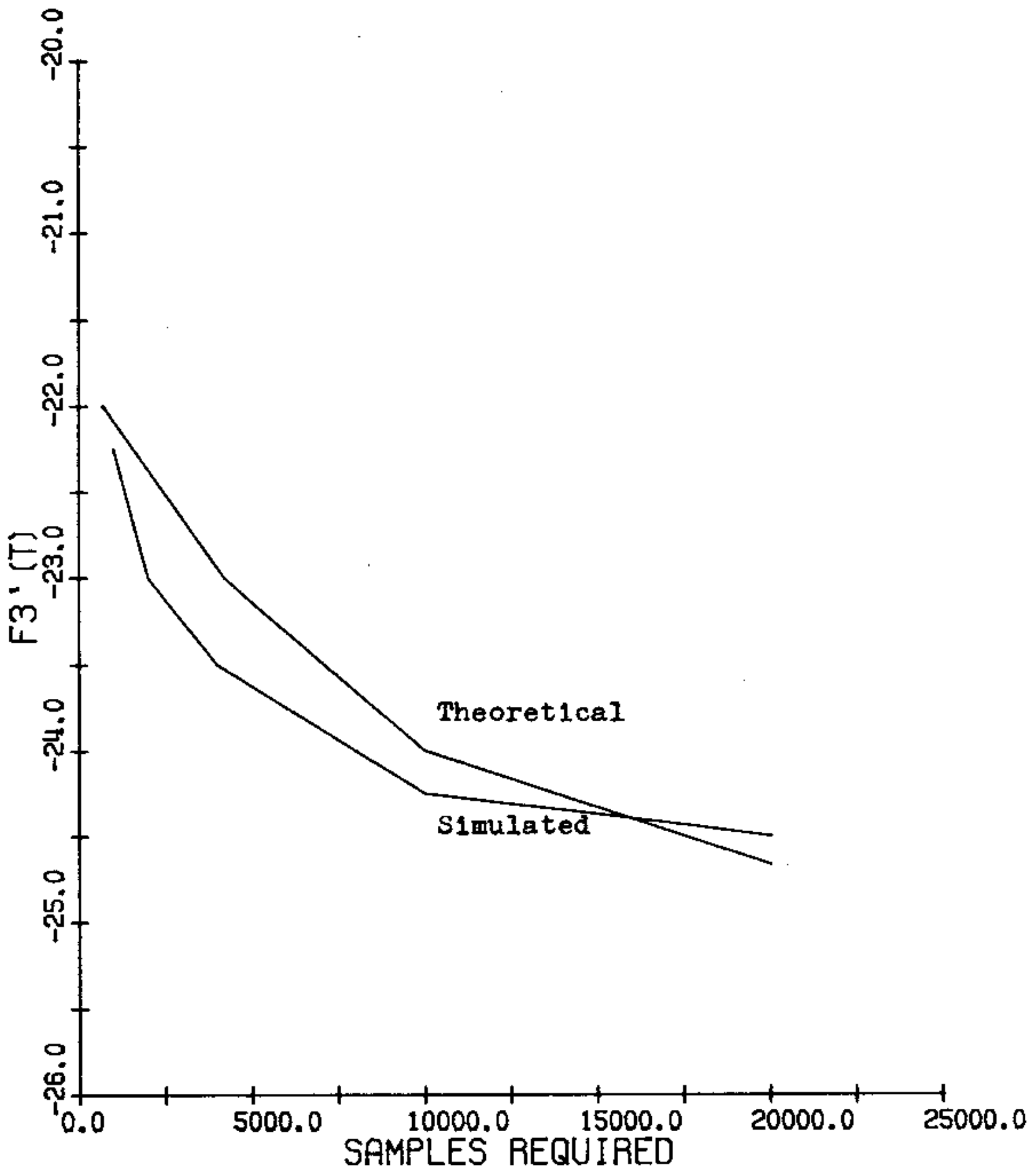


Figure B.3a: **Off-line performance curves for random search on test function F3.**

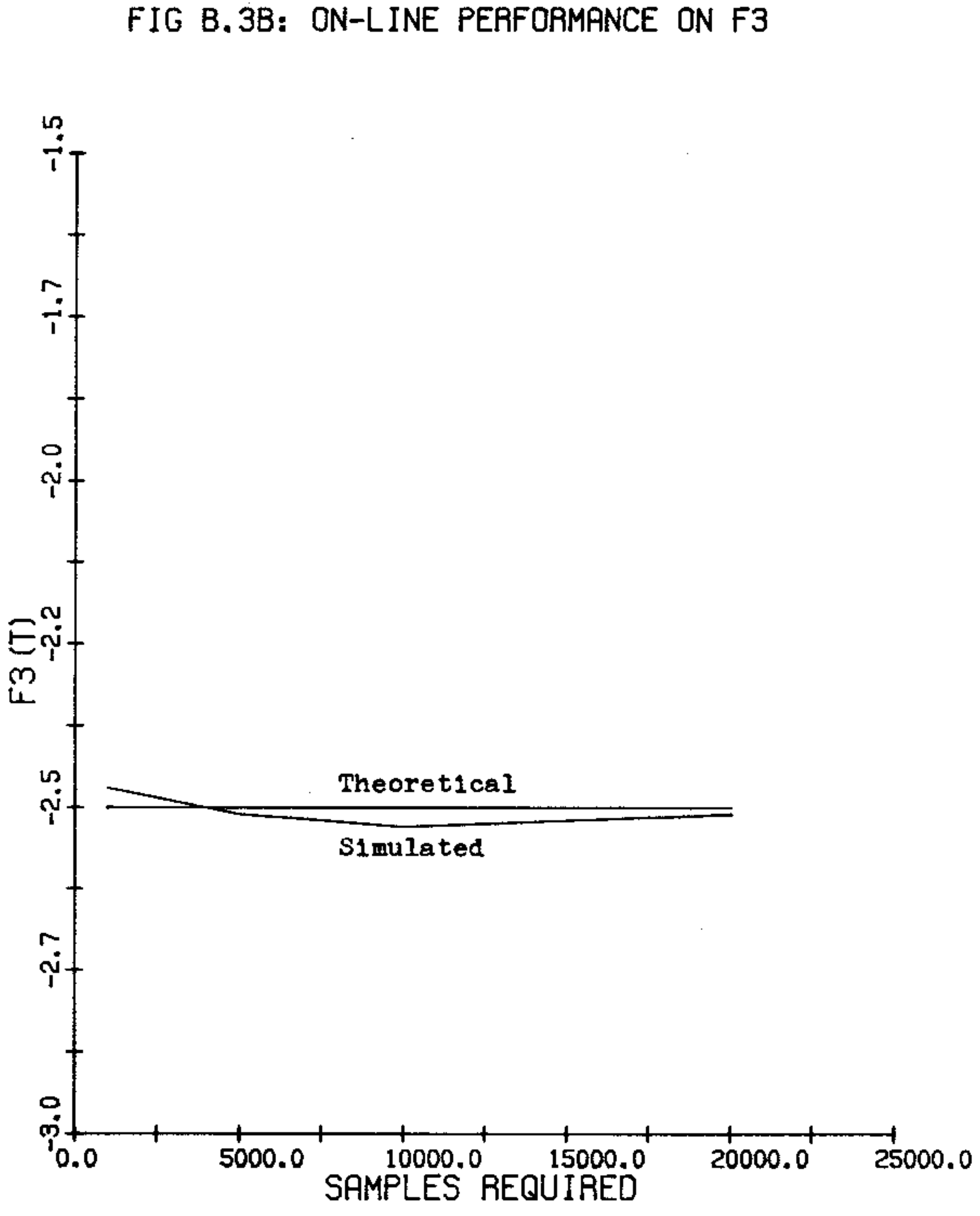


Figure B.3b: On-line performance curves for random search on test function F3.

B.6 RANDOM on F4

Deriving expected off-line performance for F4 is difficult. We need to compute PROB$\left[\sum_{1}^{30} ix_i^4+\text{GAUSS}(0,1)\leq\epsilon\right]$. The problem can be simplified somewhat by noting that over the interval of observation, RANDOM was unable to reduce F4*(t) below 50. This suggests that a reasonable approximation is obtained by ignoring the Gaussian noise, that is, PROB$\left[\sum_{1}^{30} ix_i^4\leq\epsilon\right]$. As in the case of F1, we have that

$$\text{PROB}\left[\sum_{1}^{30} ix_i^4\leq\epsilon\right]=\frac{V_s\left(\epsilon^{1/4}\right)}{V_c(2b)}\,,\quad \epsilon^{1/4}\leq b$$

Note, however, that b=1.28 for F4, limiting the above computation to $\epsilon<3$ which is far beyond any practical interval of observation.

Alternatively, we can attempt to bound the PROB$\left[\sum_{1}^{30} ix_i^4\leq\epsilon\right]$ by noting that:

$$\left\{X:\ |x_i|\leq\left(\frac{\epsilon}{30i}\right)^{1/4}\right\}\subset\left\{X:\ \sum_{1}^{30} ix_i^4\leq\epsilon\right\}\subset\left\{X:\ |x_i|\leq\left(\frac{\epsilon}{i}\right)^{1/4}\right\}$$

However, because of the high dimensionality of F4, these bounds are extremely crude, effectively bounding the probability by 0 and 1. As a consequence no direct validation of the off-line performance curve for RANDOM on F4 (shown in figure B.4a) was made. Rather, an attempt was made to minimize any bias due to pseudo-randomness by rotating and inverting the space as the performance

FIG. B.4A: OFF-LINE PERFORMANCE ON F4

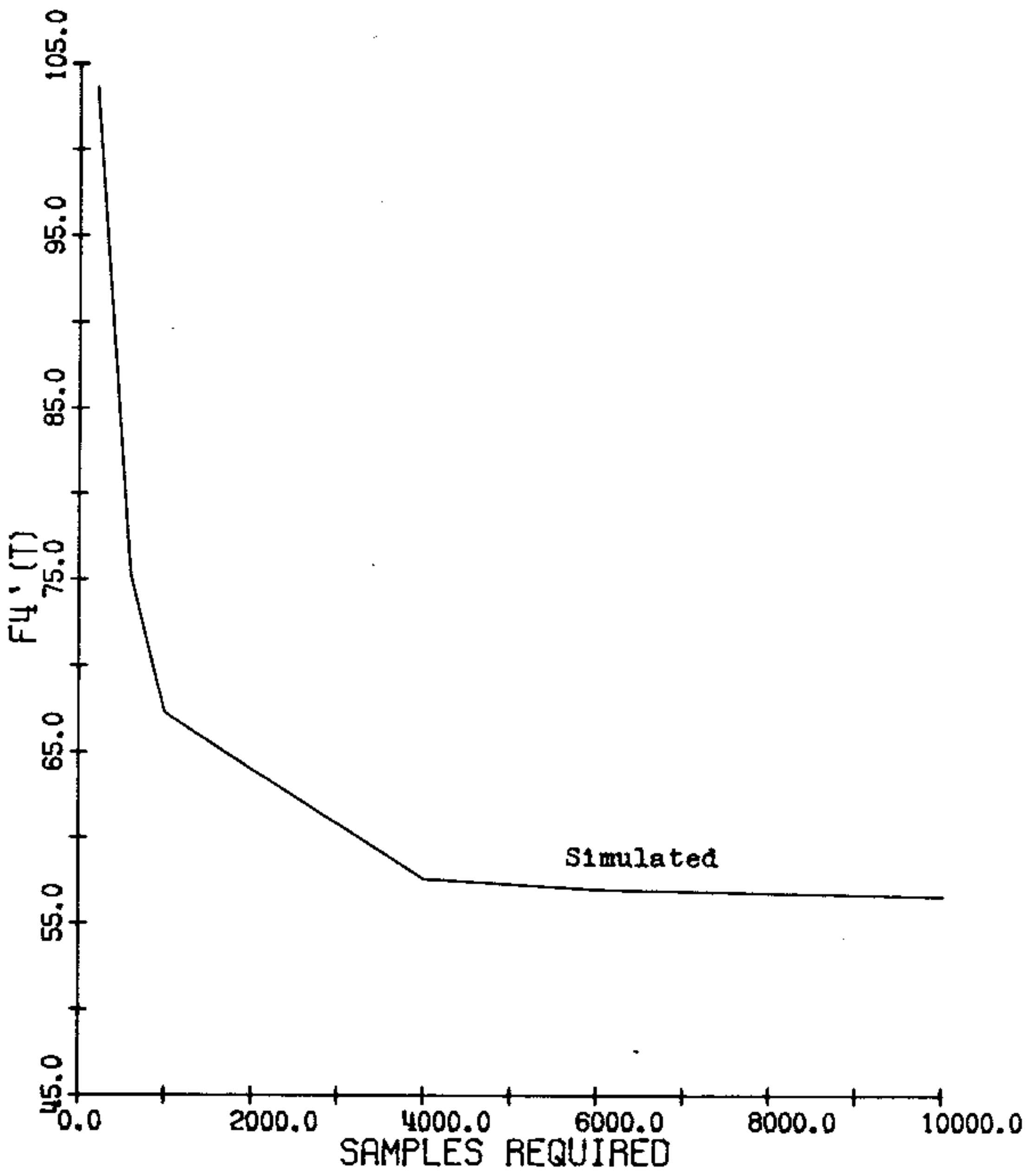


Figure B.4a: Off-line performance curves for random search on test function F4.

curve was generated.

Expected on-line performance on F4 is given by:

$$\mathrm{AVE}(F4(X)) = \frac{1}{(2b)^{30}} \int\int_{-b}^{b}\cdots\int \sum_{1}^{30} ix_i^4 \, dx_1 \ldots dx_{30}$$

$$= \frac{1}{(2b)^{30}} \sum_{1}^{30} \int\int_{-b}^{b}\cdots\int ix_i^4 \, dx_1 \ldots dx_{30}$$

$$= \frac{1}{(2b)^{30}} \sum_{1}^{30} i \cdot \frac{b^4}{5} \cdot (2b)^{30}$$

$$= \frac{b^4}{5} \sum_{1}^{30} i$$

$$= 93b^4$$

Figure B.4b compares the expected on-line performance of random search on F4 with the curve generated by RANDOM. The results closely match.

B.7 RANDOM on F5

To analytically derive the expected off-line performance for random search on F5 is difficult. However, because of the composite nature of F5, a reasonable approximation is not difficult. Since near the j th local minimum $F5(X) \cong f_j(X)$ and elsewhere $F5(X) \cong 500$, we have

FIG. B.4B: ON-LINE PERFORMANCE ON F4

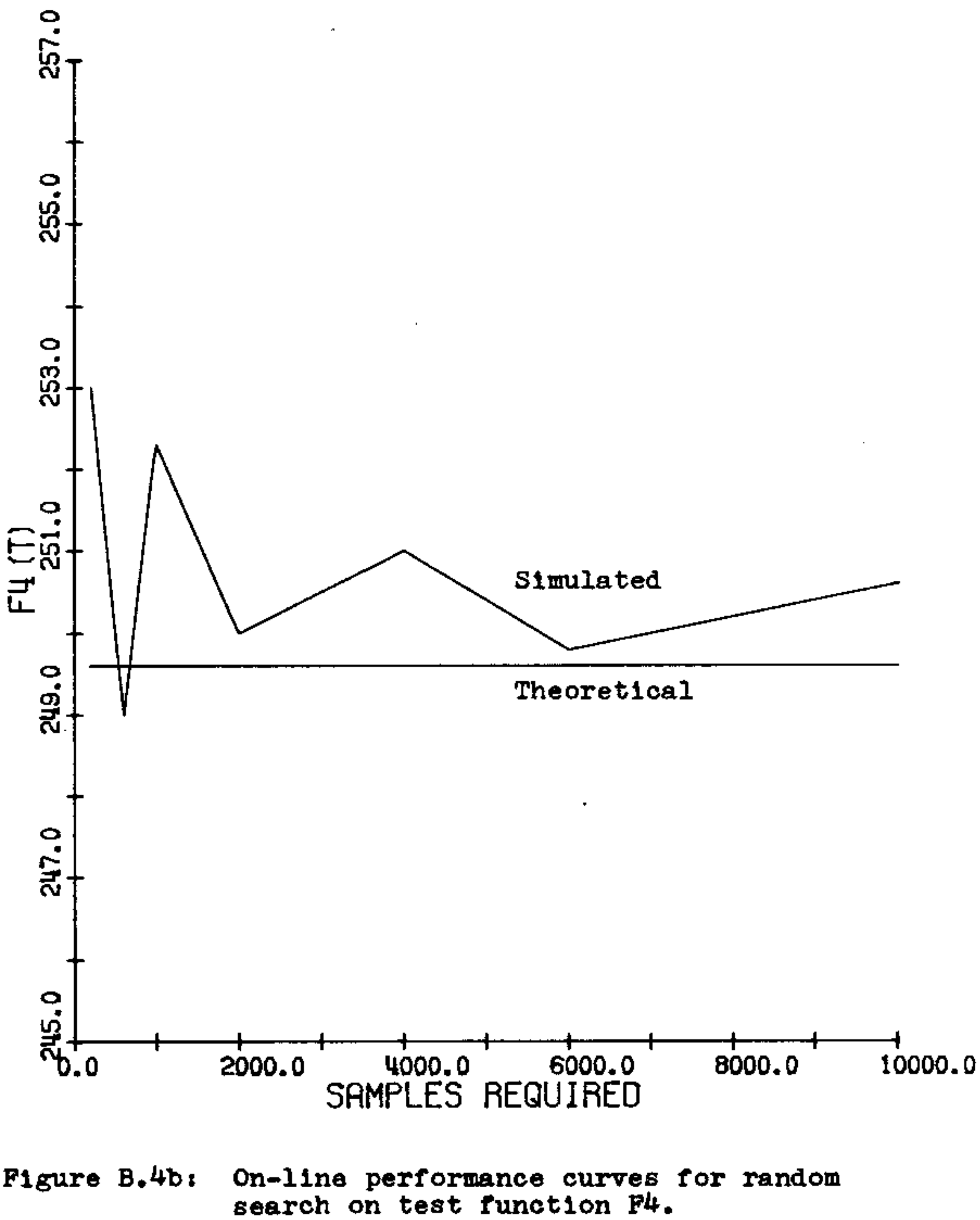


Figure B.4b: On-line performance curves for random search on test function F4.

$$\text{PROB}\left[F5(X) \leq \epsilon\right] = \sum_{j=1}^{25} \text{PROB}\left[f_j(X) \leq \epsilon\right] \quad , \ \epsilon << 500$$

For each j we have:

$$\text{PROB}\left[f_j(X) \leq \epsilon\right] = \text{PROB}\left[c_j + \sum_{i=1}^{2} (x_i - a_{ij})^6 \leq \epsilon\right]$$

$$= \text{PROB}\left[\sum_{i=1}^{2} (x_i - a_{ij})^6 \leq \epsilon - c_j\right]$$

$$= \text{PROB}\left[\sum_{i=1}^{2} x_i^6 \leq \epsilon - c_j\right]$$

$$= \begin{cases} 0 \text{ if } \epsilon < c_j \\ \dfrac{V_s((\epsilon - c_j)^{1/6})}{V_c(2b)} \text{ if } 0 \leq (\epsilon - c_j)^{1/6} \leq b \end{cases}$$

where

$$V_s(k^{1/6}) = \int_{-k^{1/6}}^{k^{1/6}} \int_{-(k-x_2^6)^{1/6}}^{(k-x_2^6)^{1/6}} 1 \ dx_1 dx_2$$

$$= 4 \int_{0}^{k^{1/6}} (k - x_2^6)^{1/6} \ dx_2$$

$$= 4k^{1/3} \int_0^1 (1-y^6)^{1/6}\, dy$$

which is beyond my powers of integration. However, numerical integration yields:

$$\int_0^1 (1-y^6)^{1/6}\, dy = .963688$$

If we assume this is accurate enough for our purposes, we have:

$$\text{PROB}\left[f_j(X) \leq \epsilon\right] \cong \begin{cases} 0 \text{ if } \epsilon < c_j \\ \dfrac{3.854752}{(2b)^2} * (\epsilon - c_j)^{1/3}, & 0 \leq (\epsilon - c_j)^{1/6} \leq b \end{cases}$$

This approximation was used to estimate the expected off-line performance of random search on F5. Figure B.5a compares this approximation with the curve generated by RANDOM. The two agree surprisingly well.

The expected on-line performance of random search on F5 is also difficult to derive. In fact, even a reasonable approximation is difficult to find. As a consequence, no direct validation of the on-line performance curve for RANDOM shown in figure B.5b was done. Rather, the search space was rotated and inverted while

FIG. B.5A: OFF-LINE PERFORMANCE ON F5

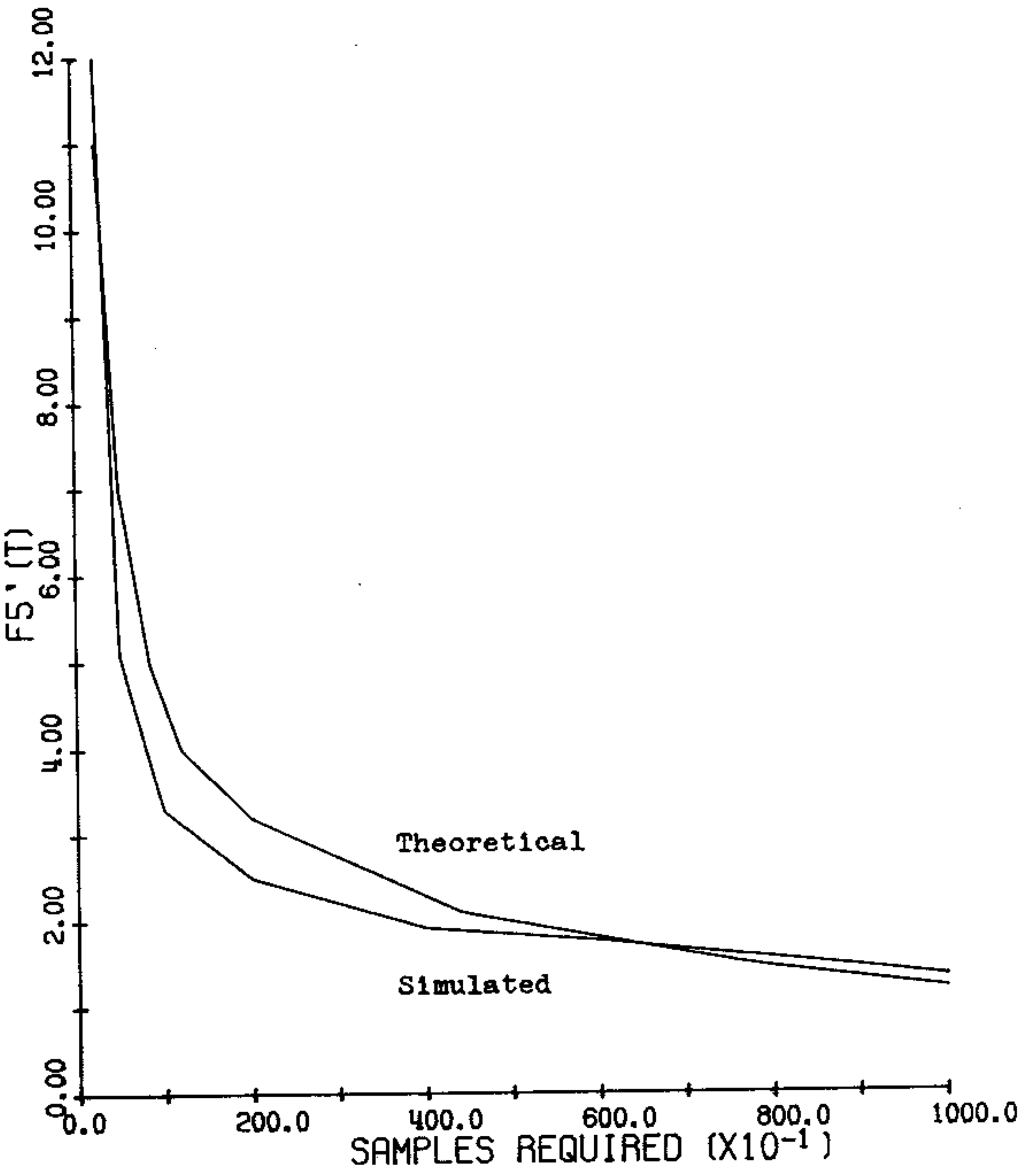


Figure B.5a: Off-line performance curves for random search on test function F5.

FIG. B.5B: ON-LINE PERFORMANCE ON F5

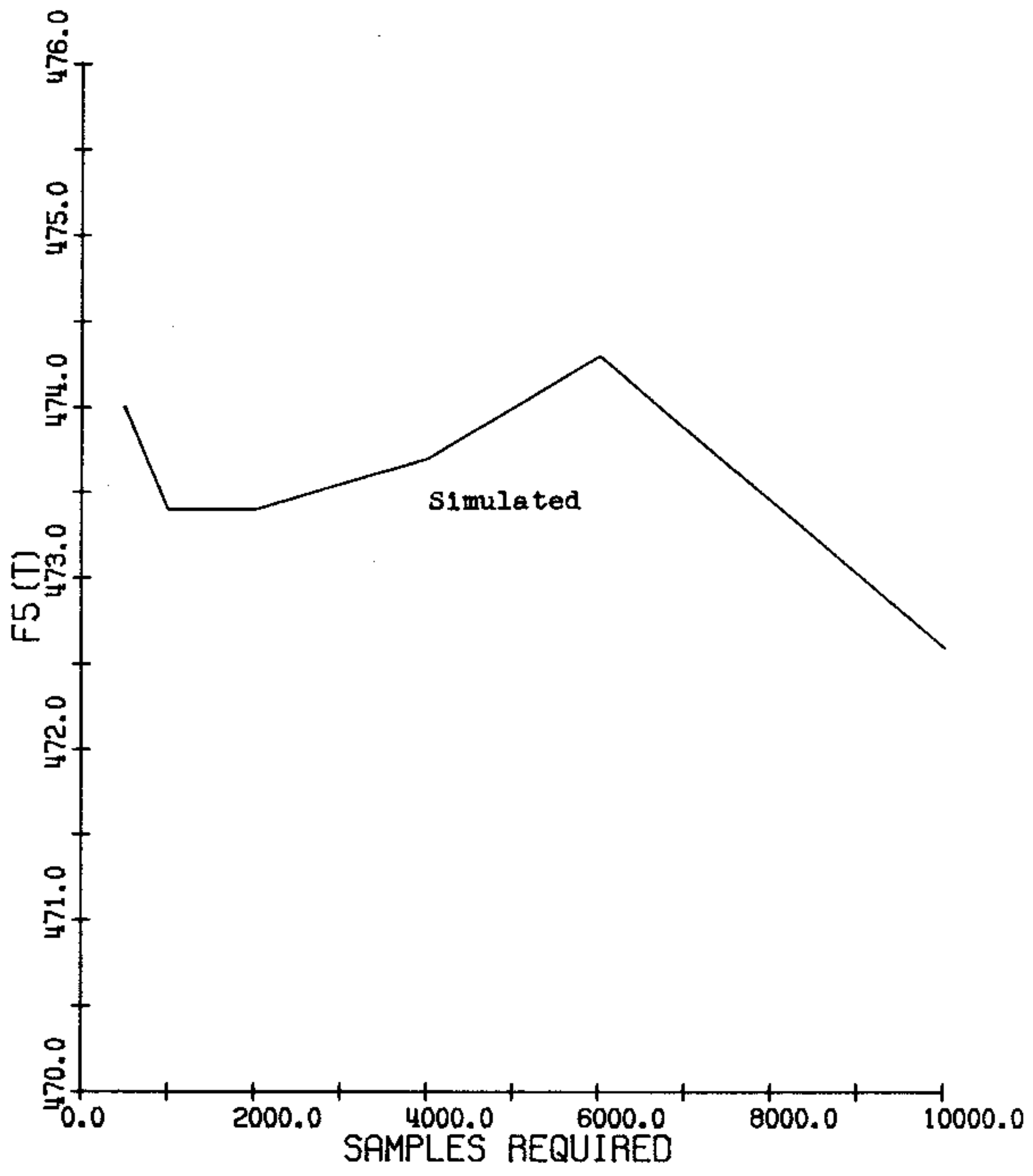


Figure B.5b: On-line performance curve for random search on test function F5.

generating the performance curve in an attempt to minimize any bias due to pseudo-randomness.

Appendix C

PLAN R1 ON E

C.1 Introduction

The idea of simulating the information processing mechanisms of heredity and evolution as strategies for adaptation in artificial systems was first introduced by Holland. Since then considerable experimentation and analysis has been done in such areas as the kind of genetic operators to be applied, the form of the representation space, the mating rules to be used, and so on.

Bagley has compared the behavior of correlation algorithms and genetic algorithms with respect to environments exhibiting first and second order non-linearities. He showed that comparable or superior behavior could be generated by genetic algorithms using population sizes of 200, standard genetic operators applied at fixed levels, and random mating.

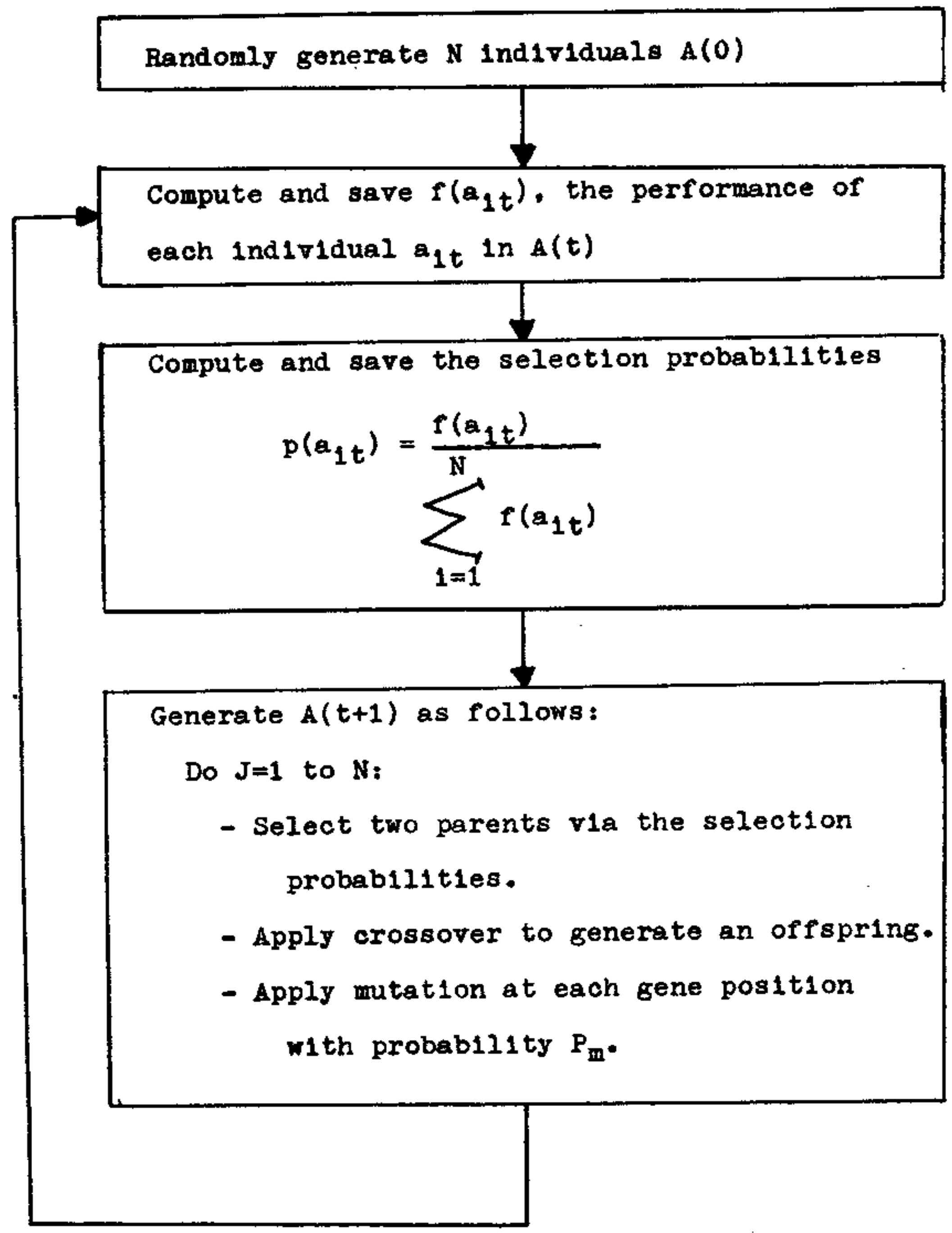
Cavicchio has studied the behavior of genetic algorithms in a pattern-matching environment. He was able to show the negative effects of a small population size (less than 20) and experimented with several forms of genetic operators applied at varying levels. He was able to generate excellent adaptive behavior using a scheme for modifying the frequency with which genetic operators are applied during adaptation.

Hollstien has studied the effects that representation and breeding plans have on the behavior of genetic algorithms in a variety of continuous and discontinuous two-variable environments. Using mutation and crossover at fixed rates and a population of size 16, he exhibited robust behavior for both random mating and recurrent inbreeding-crossbreeding. He showed the negative effects of such breeding plans as linebreeding and inbreeding. He also examined the effect of several representations for genes including binary, gray code, and polygene forms.

Frantz has studied the effects of the genetic operators on the composition of the resultant population in the face of highly non-linear environments. He was able empirically to verify hypotheses about the effects of crossover and mutation, but had difficulty in verifying inversion effects.

Foo and Bosworth have attempted mathematical analyses of the basic genetic operators. Bosworth, Foo, and Zeigler have experimented with incorporating gradient information as a mutation operator in the context of function optimization. In the same context Zeigler, Bosworth, and Bethke have shown genetic algorithms to be relatively insensitive to noisy environments when compared with classical gradient optimization techniques.

The cumulative experience of these studies provided the framework for the definition of the initial elementary reproductive plan R1 introduced in chapter 2 and which operates as follows:



Randomly generate N individuals A(0)

Compute and save $f(a_{it})$, the performance of each individual a_{it} in A(t)

Compute and save the selection probabilities

$$p(a_{it}) = \frac{f(a_{it})}{\sum_{i=1}^{N} f(a_{it})}$$

```
Generate A(t+1) as follows:
  Do J=1 to N:
    - Select two parents via the selection
        probabilities.
    - Apply crossover to generate an offspring.
    - Apply mutation at each gene position
        with probability $P_m$.
```

This actually specifies a class of genetic algorithms which differ in the parameters N and P_m. Previous experience suggested that population sizes smaller than 30 were subject to severe stochastic effects which made performance measurements difficult. For these measurements, a population size of N=50 was chosen, and a mutation rate of .001 which is an upperbound on the estimate of the mutation rate in nature.

Because R1 is a stochastic process, a minimum of 5 trials was made on each test function. More were made if required to reduce the standard 95% confidence intervals associated with the performance measurements. The results are presented in several ways. Graphs are given to compare the off-line and on-line performance curves $f^*(t)$ and $f(t)$ of R1 and random search on each of the test functions described in appendix A. Tables are given to compare the performance indices $x_e^*(T)$ and $x_e(T)$ for R1 and random search for various values of T on each of the test functions. Finally, the robustness of R1 and random search on E is compared using the measures defined in Chapter 1.

C.2 <u>Plan R1 on F1</u>

Figures C.1a and C.1b compare the performance curves for R1 and random search on F1 over the interval of observation. The associated performance ratings $x_{F1}^*(T)$ and $x_{F1}(T)$ are tabularized below for various values of T:

FIG. C.1A: OFF-LINE PERFORMANCE ON F1

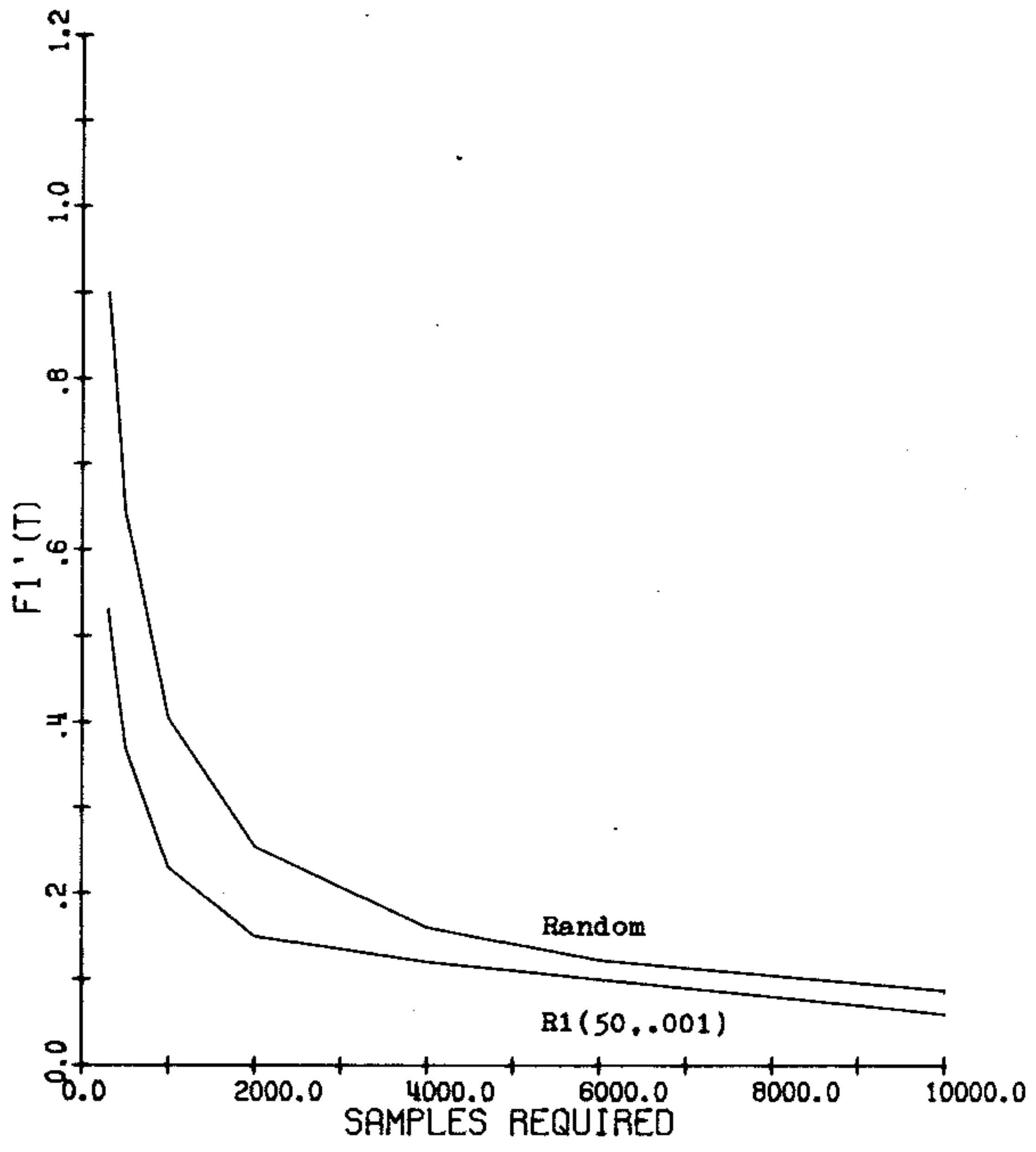


Figure C.1a: Off-line performance curve for plan R1 on test function F1.

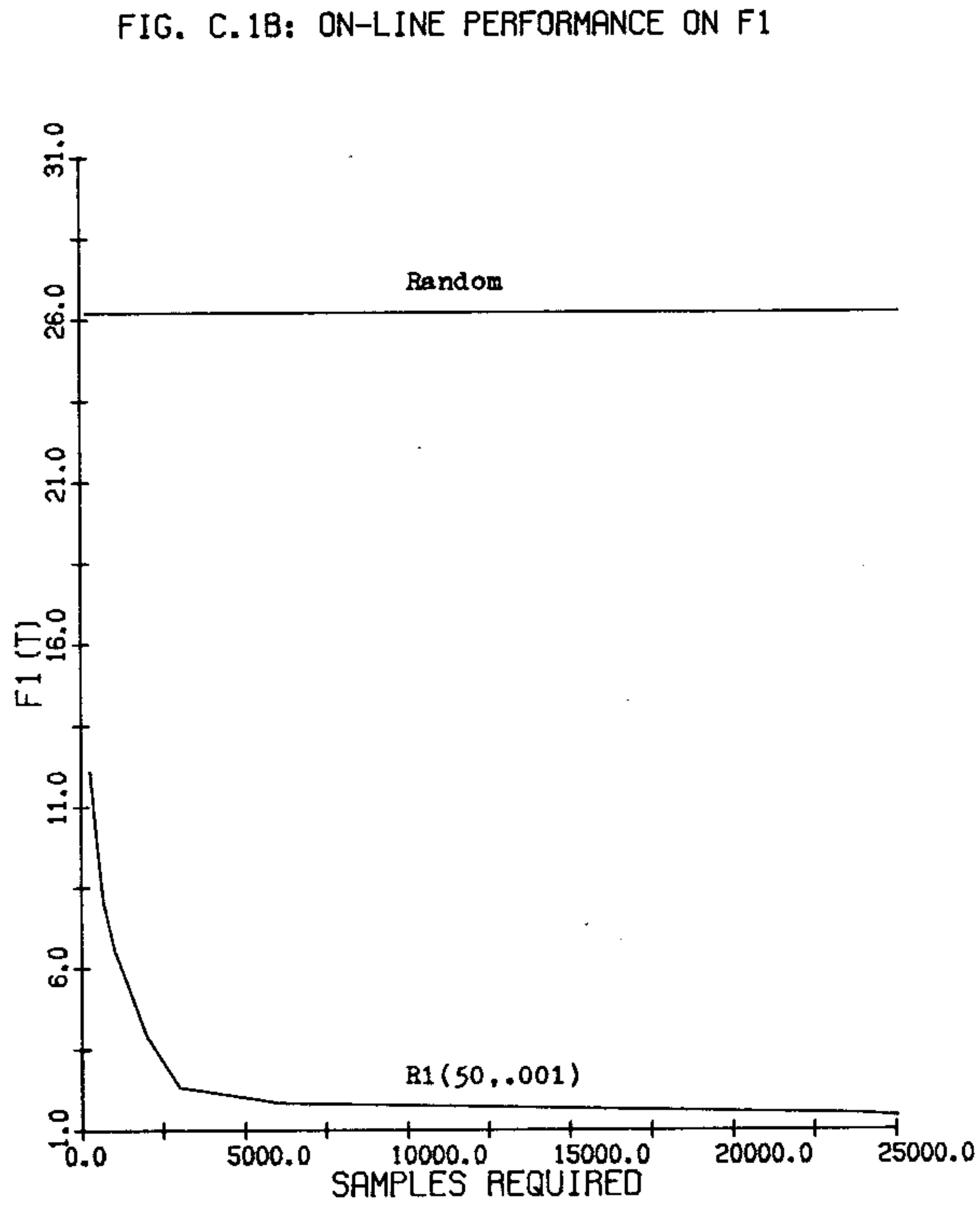


Figure C.1b: On-line performance curve for plan R1 on test function F1.

$x^*_{F1}(T)$

T	R1	Random
500	1.25	3.0
1000	.73	.94
2000	.48	.75
4000	.28	.44
6000	.22	.36
10000	.125	.27

$x_{F1}(T)$

T	R1	Random
500	14.0	26.2
1000	10.6	26.2
2000	7.8	26.2
4000	5.3	26.2
6000	4.6	26.2
10000	3.1	26.2

So we see that R1 performs better than random search on F1 over the interval of observation. This, of course, was expected for on-line performance. Notice that the off-line performance of R1 is not strikingly better; it gets off to a good start and then seems to plateau. This observation is explored more fully in chapter 3.

C.3 Plan R1 on F2

Figures C.2a and C.2b compare the performance curves of R1 and random search on F2. The associated performance

FIG. C.2A: OFF-LINE PERFORMANCE ON F2

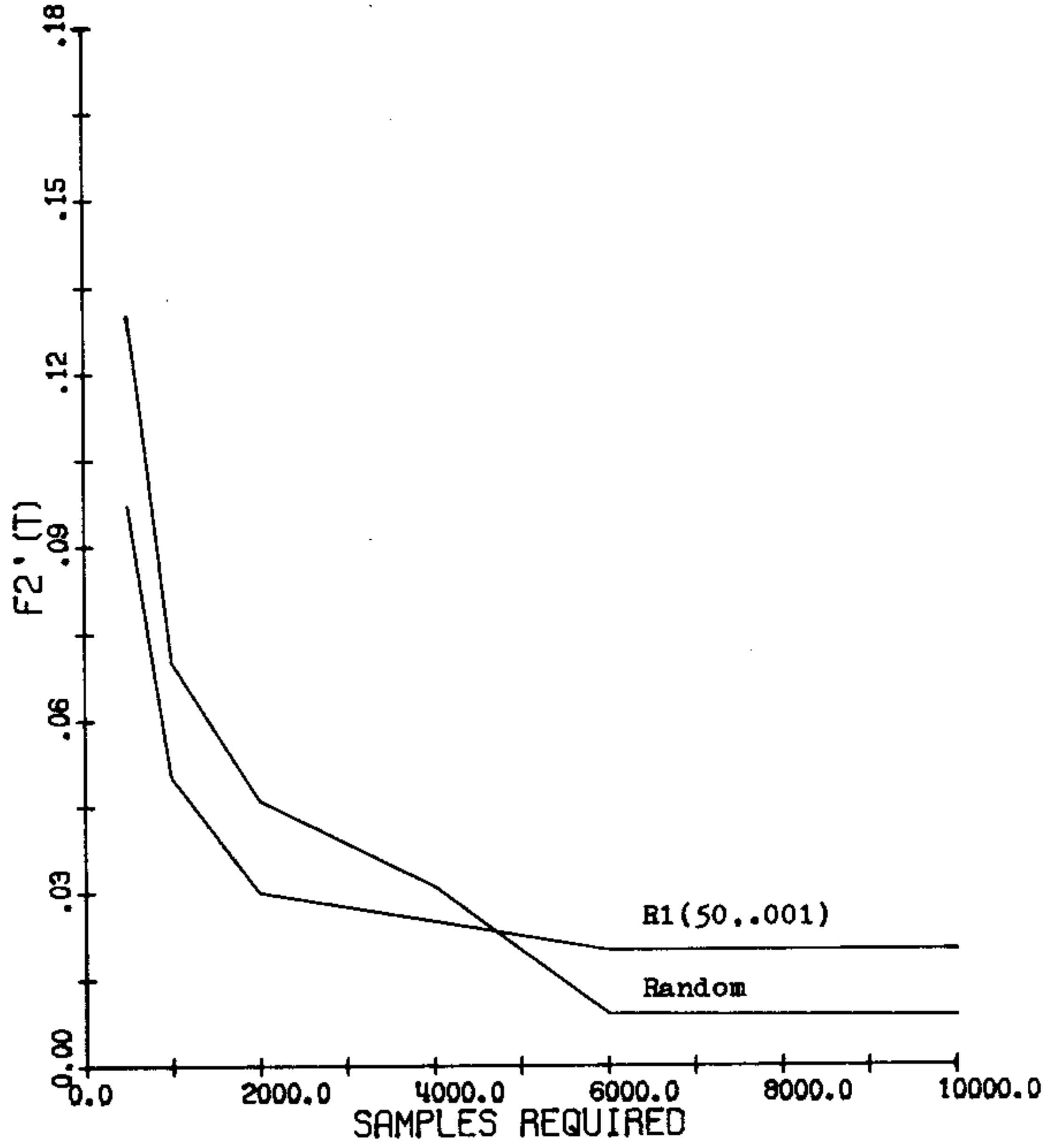


Figure C.2a: Off-line performance curve for plan R1 on test function F2.

FIG. C.2B: ON-LINE PERFORMANCE ON F2

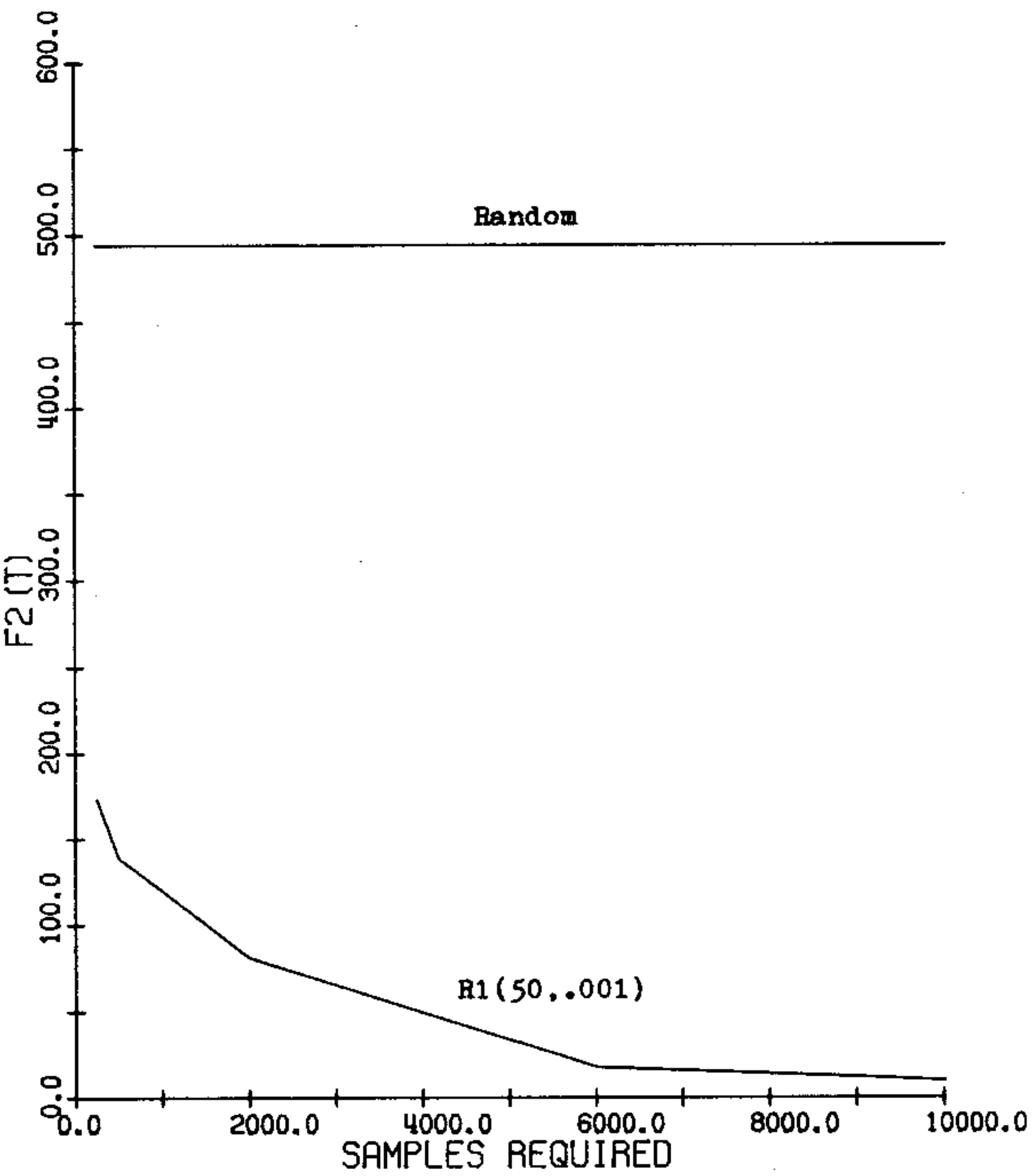


Figure C.2b: On-line performance curve for plan R1 on test function F2.

ratings $x^*_{F2}(T)$ and $x_{F2}(T)$ are tabularized below for various values of T:

$x^*_{F2}(T)$

T	R1	Random
500	3.72	3.97
1000	1.84	1.93
2000	.93	1.02
4000	.47	.53
6000	.34	.35
10000	.25	.20

$x_{F2}(T)$

T	R1	Random
500	219.3	494.05
1000	170.0	494.05
2000	133.3	494.05
4000	100.2	494.05
6000	78.4	494.05
10000	67.7	494.05

So we see that R1 generally outperforms random search on F2 over the interval of observation. This, of course, was expected for on-line performance. However, note that off-line performance is initially better, but then actually falls behind random search. This observation is explored more fully in chapters 3 and 4.

C.4 Plan R1 on F3

Figures C.3a and C.3b compare the performance curves of R1 and random search on F3. The associated performance ratings $x^*_{F3}(T)$ and $x_{F3}(T)$ are tabulated below for various values of T:

$x^*_{F3}(T)$

T	R1	Random
500	-23.1	-18.0
1000	-23.8	-19.7
2000	-24.6	-21.1
4000	-25.5	-22.3
6000	-26.2	-22.7
10000	-27.1	-23.3

$x_{F3}(T)$

T	R1	Random
500	-10.8	-2.5
1000	-14.4	-2.5
2000	-18.3	-2.5
4000	-21.0	-2.5
6000	-22.1	-2.5
10000	-23.0	-2.5

So we see that R1 performed considerably better on F3 than random search, indicating that discontinuous functions pose no problem for genetic plans.

FIG C.3A: OFF-LINE PERFORMANCE ON F3

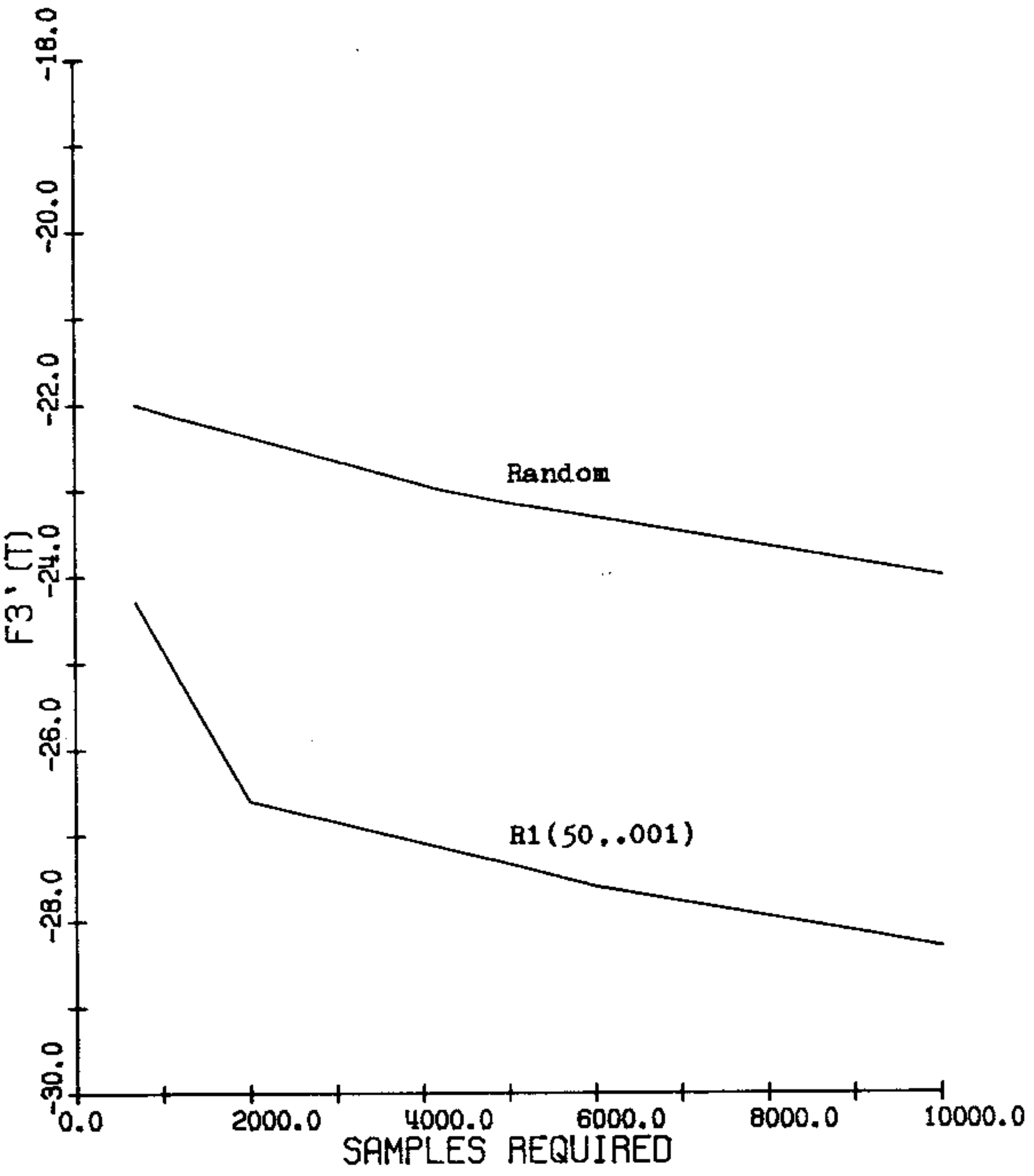


Figure C.3a: Off-line performance curve for plan R1 on test function F3.

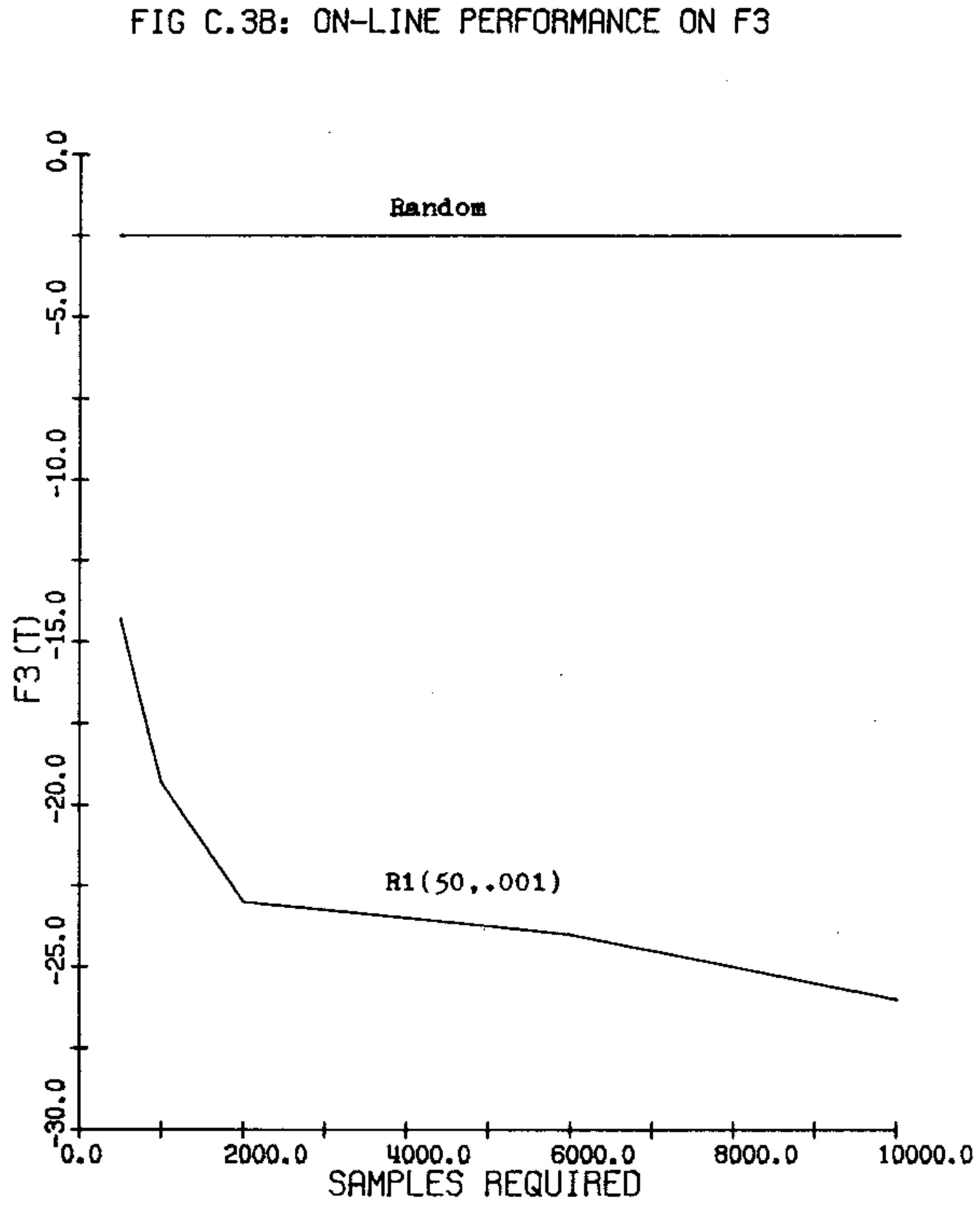


Figure C.3b: On-line performance curve for plan R1 on test function F3.

C.5 Plan R1 on F4

Figures C.4a and C.4b compare the performance curves for R1 and random search on F4. The associated performance indices computed for several values of T are given below:

$x^*_{F4}(T)$

T	R1	Random
500	80.9	105.0
1000	61.2	89.2
2000	46.7	78.6
4000	38.5	70.6
6000	35.9	66.3
10000	31.6	63.7

$x_{F4}(T)$

T	R1	Random
500	180.1	249.6
1000	149.5	249.6
2000	120.9	249.6
4000	105.4	249.6
6000	97.3	249.6
10000	75.6	249.6

So we see that R1 performed considerably better on F4 than random search, indicating that high-dimensionality and noise pose no particular problems for genetic algorithms.

FIG. C.4A: OFF-LINE PERFORMANCE ON F4

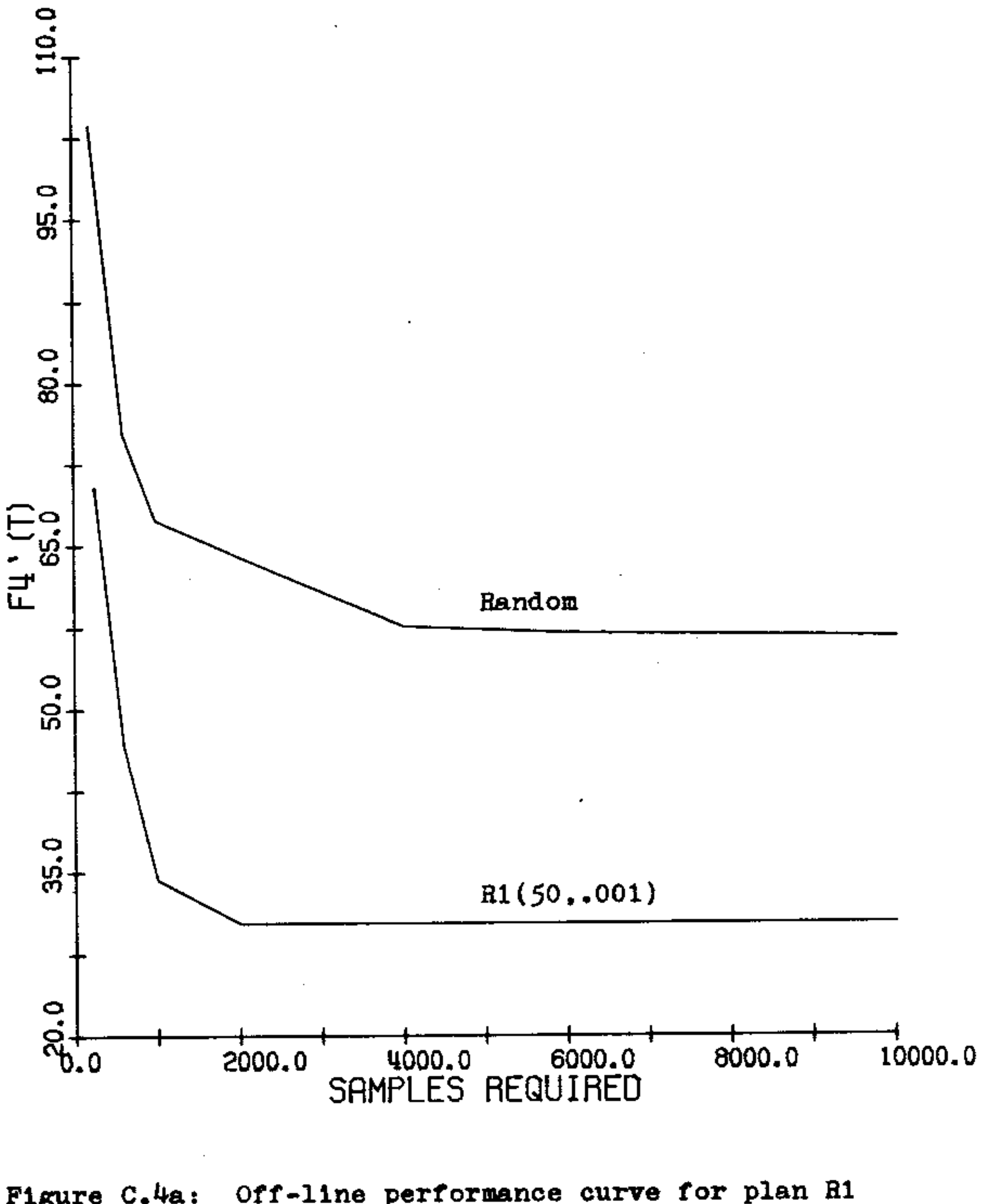


Figure C.4a: Off-line performance curve for plan R1 on test function F4.

FIG. C.4B: ON-LINE PERFORMANCE ON F4

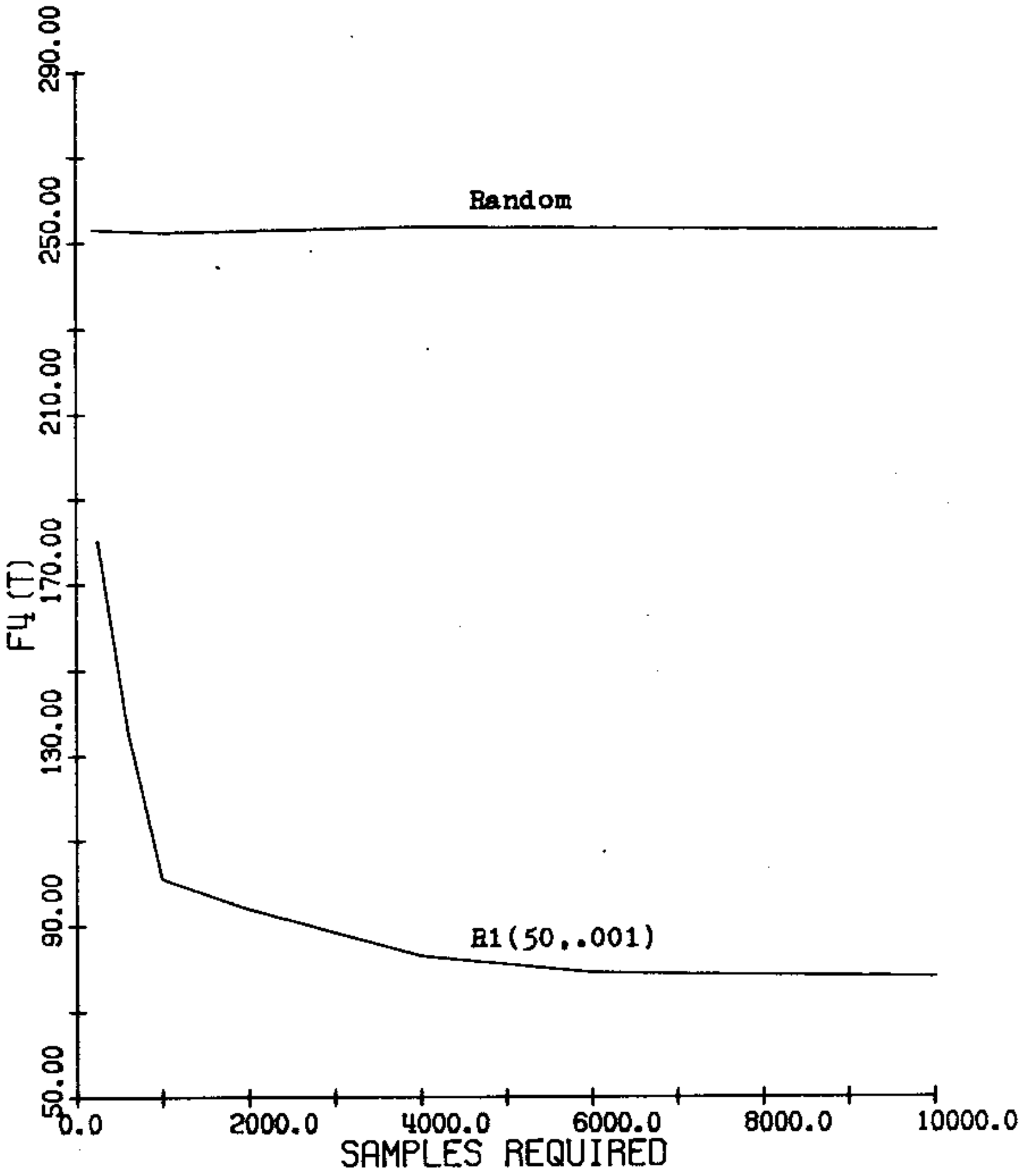


Figure C.4b: On-line performance curve for plan R1 on test function F4.

C.6 Plan R1 on F5

Figures C.5a and C.5b compare the performance curves for R1 and random search on F5. The associated performance indices computed for various values of T are given below:

$x^*_{F5}(T)$

T	R1	Random
500	11.4	35.7
1000	6.21	19.2
2000	3.74	10.6
4000	2.15	6.25
6000	1.83	4.82
10000	1.61	3.45

$x_{F5}(T)$

T	R1	Random
500	127.0	474
1000	99.7	473.4
2000	71.9	473.4
4000	51.6	473.7
6000	36.2	473.3
10000	14.7	472.6

So we see that R1 outperforms random search on F5, indicating that multimodal surfaces pose no particular problems for genetic algorithms.

C.7 Robustness of Plan R1

Robustness is defined in chapter 1 as the average

FIG. C.5A: OFF-LINE PERFORMANCE ON F5

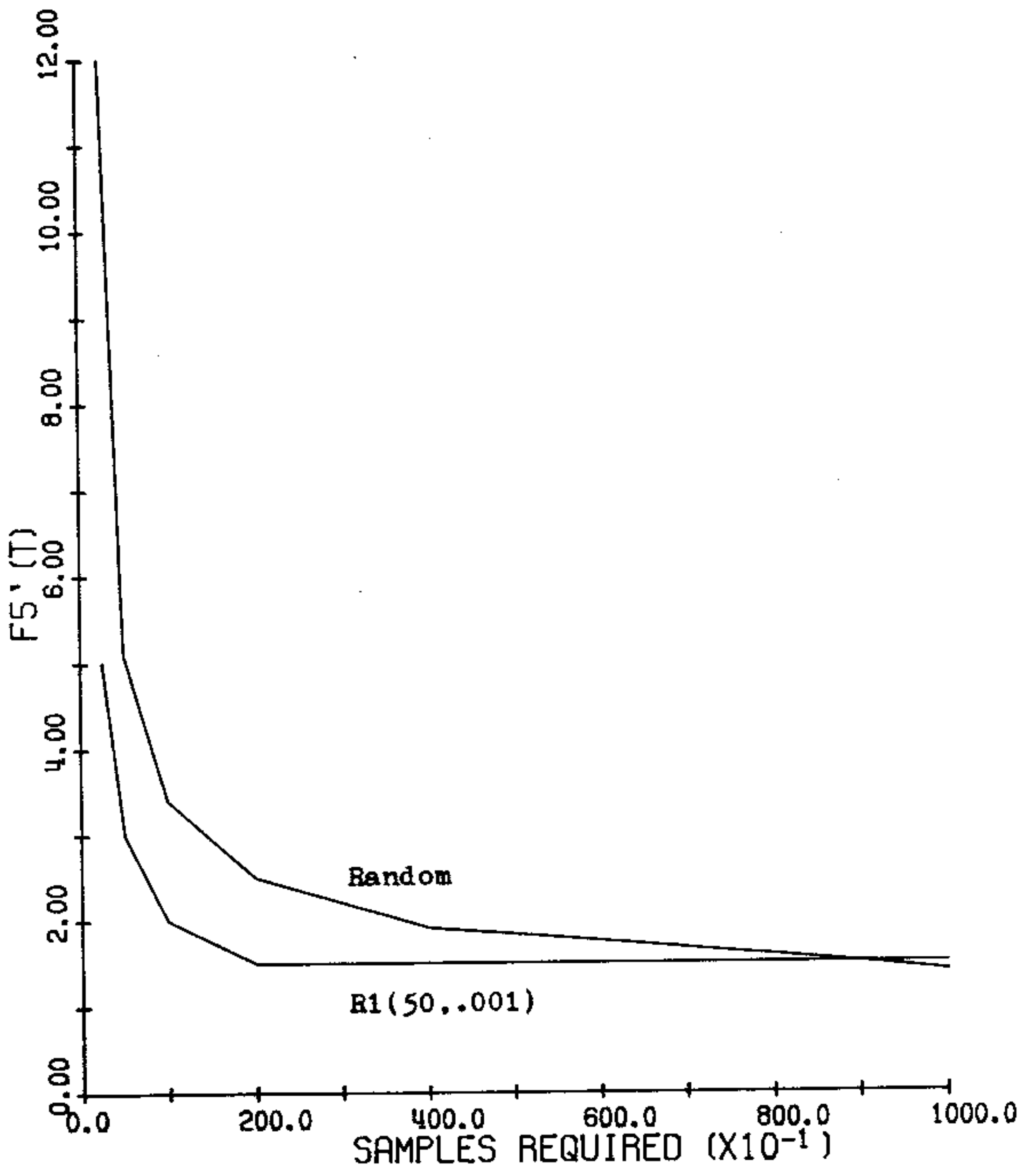


Figure C.5a: Off-line performance curve for plan R1 on test function F5.

FIG. C.5B: ON-LINE PERFORMANCE ON F5

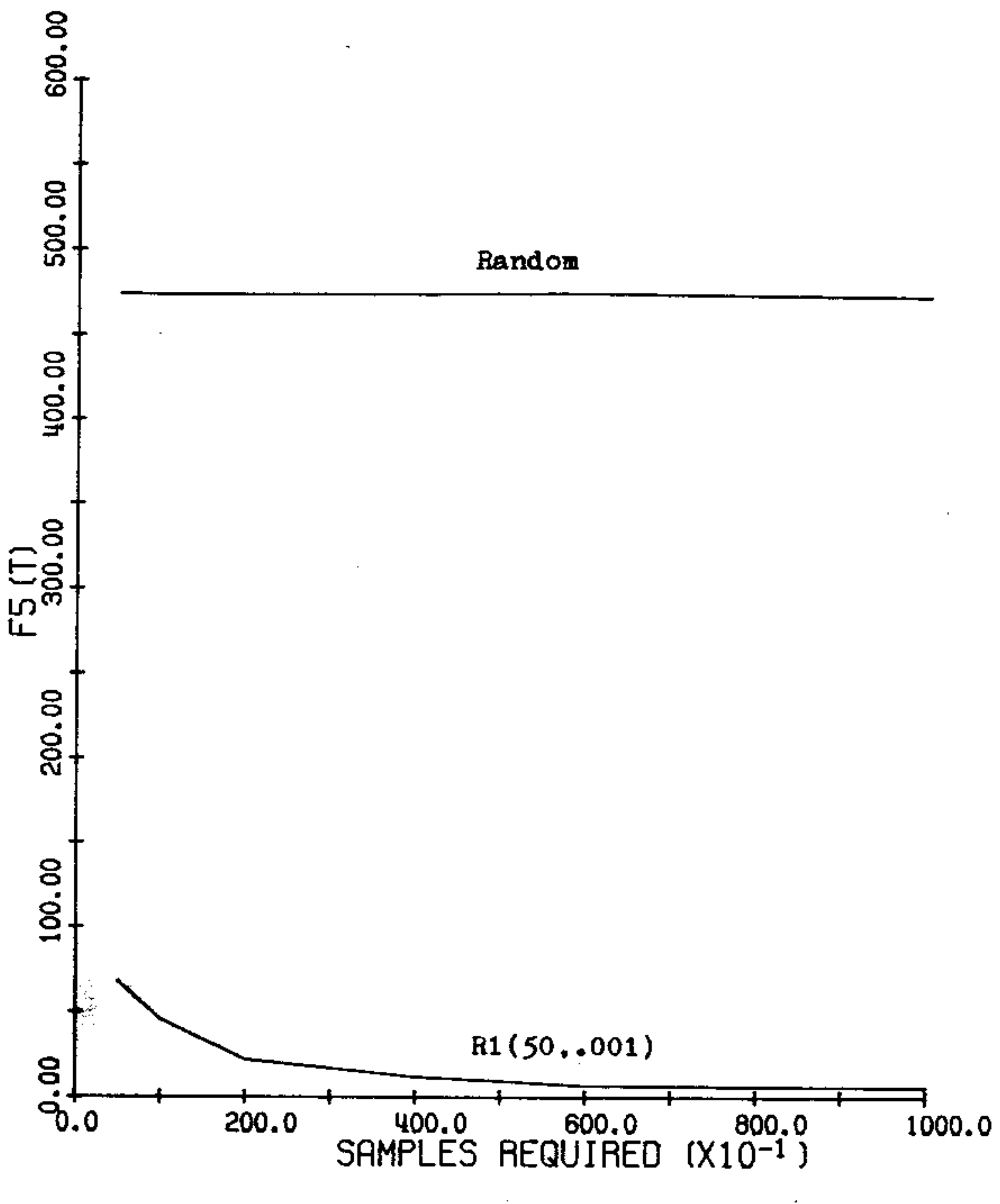


Figure C.5b: On-line performance curve for plan R1 on test function F5.

performance rating over E. This is computed for various values of T below:

$X^*_E(T)$

T	H1	Random
500	14.83	25.93
1000	9.24	18.34
2000	5.45	13.97
4000	3.18	11.10
6000	2.48	9.83
10000	1.29	8.86

$X_E(T)$

T	H1	Random
500	105.92	147.66
1000	83.08	147.77
2000	63.12	147.57
4000	48.30	147.59
6000	38.88	147.61
10000	27.62	147.55

So we see that H1 is definitely more robust on E than random search is.

BIBLIOGRAPHY

Aigner, D. J. 1968. Principles of Statistical Decision Making. New York: MacMillan.

Bagley, J. D. 1967. The behavior of adaptive systems which employ genetic and correlation algorithms. Doctoral Thesis, Department of Computer and Communication Sciences, University of Michigan.

Bauer, M. J. 1974. A simulation approach to the design of dynamic feedback scheduling algorithms for time-shared computer systems. Simuletter, ACM SIGSIM Quarterly. 514: 23-31.

Bellman, R. 1959. Adaptive Control Processes: A Guided Tour. Princeton University Press.

Bosworth, J. L.; Foo, N.; and Zeigler, B. P. 1972. Comparison of genetic algorithms with conjugate gradient methods. University of Michigan Technical Report No. 00312-1-T.

Bremermann, H. 1970. A method of unconstrained global optimization. Mathematical Biosciences. 9:1-15.

Brent, R. P. 1971. Algorithms for finding zeros and extrema of functions without calculating derivatives. Doctoral Thesis, Computer Science Department, Stanford.

Cavicchio, D. J. Jr. 1970. Adaptive search using simulated evolution. Doctoral Thesis, Department of Computer and Communication Sciences, University of Michigan.

Crow, J. F., and Kimura, M. 1970. An Introduction to Population Genetics Theory. Harper & Row.

Feller, W. 1950. An Introduction to Probability Theory and its Applications, Volume I. New York: John Wiley & Sons.

Fletcher, R. and Powell, M. J. D. 1963. A rapidly convergent descent method for minimization. The Computer Journal. 6: 163.

Fletcher, R. and Reeves, C. M. 1964. Function minimization by conjugate gradients. The Computer Journal. 7: 149.

Fletcher, R. 1970. A new approach to variable metric algorithms. The Computer Journal. 13: 317.

Foo, N. Y. and Bosworth, J. L. 1972. Algebraic, geometric, and stochastic aspects of genetic operators. University of Michigan Technical Report No. 003120-2-T.

Frantz, D. R. 1972. Non-linearities in genetic adaptive search. Doctoral Thesis, Department of Computer and Communication Sciences, University of Michigan.

Hill, J. D. 1969. A search technique for multimodal surfaces. IEEE Transactions on System Science and Cybernetics, SSC-3,1.

Hogg, R. V. and Craig, A. T. 1965. Introduction to Mathematical Statistics. New York: MacMillan.

Holland, J. H. 1962. Outline for a logical theory of adaptive systems. Journal of the Association for Computing Machinery (ACM). 3: 297-314.

________. 1967. Nonlinear environments permitting efficient adaptation. Computer and Information Sciences-II. New York: Academic Press. 147-164.

________. 1975. Adaptation in Natural and Artificial Systems. University of Michigan Press.

Hollstien, R. B. 1971. Artificial genetic adaptation in computer control systems. Doctoral Thesis, Department of Computer and Communication Sciences, University of Michigan.

Howard, R. A. 1971. Dynamic Probabilistic Systems, Volume I. New York: John Wiley & Sons.

Huang, H. Y. 1970. Unified approach to quadratically convergent algorithms for function minimization. Journal of Optimization Theory and Applications. 5: 405.

Jacoby, S. L. S.; Kowalik, J. S.; and Pizzo, J. T. 1972. Iterative Methods for Non-linear Optimization Problems. Englewood Cliffs, New Jersey: Prentice-Hall.

John, P. W. M. 1971. Statistical Design and Analysis of Experiments. New York: MacMillan.

Marsaglia, G. 1968. Random numbers fall mainly in the planes. Proc. Nat. Acad. Sci. 61,1: 25-28.

Mettler, L. E. and Gregg, T. G. 1969. Population Genetics and Evolution. Foundations of Modern Genetics Series. Prentice-Hall.

Powell, M. J. 1964. An efficient method for finding the minimum of a function of several variables without calculating derivatives. The Computer Journal. 7: 155.

Rastrigin, L. A. 1960. External control by the method of random scanning. Automatika i Telemakhanika. 21: 1264-1271.

Rosenbrock, H. H. 1960. An automatic method for finding the greatest or least value of a function. The Computer Journal. 3: 175.

Schumer, M. A. and Steiglitz, K. 1968. Adaptive step size random search. IEEE Transactions of Automatic Control. AC-13: 270.

Shekel, J. 1971. Test functions for multimodal search techniques. Fifth Annual Princeton Conference on Information Science and Systems.

Summerville, D. M. Y. 1958. An Introduction to the Geometry of N Dimensions. New York: Dover.

Sworder, D. 1966. Optimal Adaptive Control Systems. New York: Academic Press.

Tausworthe, R. C. 1965. Random numbers generated by linear recurrence modulo two. Math. Comput. 19. 90: 201-209.

Toothill, J. P. R.,; Robinson, W. D.; and Adams, A. G. 1971. The runs up-and-down performance of Tausworthe pseudo-random number generators. J. ACM 18. 3: 381-399.

Toothill, J. P. R.; Robinson, W. D.; and Eagle, D. J. 1973. An asymptotically random Tausworthe sequence. J. ACM 20. 3: 469-481.

Tsypkin, Y. Z. 1971. Adaptation and Learning in Automatic Systems. New York: Academic Press.

Wilde, D. J. and Beightler, C. S. 1969. Foundations of Optimization. Englewood Cliffs, New Jersey: Prentice-Hall.

Yahowitz, S. J. 1969. Mathematics of Adaptive Control Processes. New York: American Elsevier.

Zeigler, B. P; Bosworth, J. L.; and Bethke, A. D. 1973. Noisy function optimization by genetic algorithms and conjugate gradient methods. Logic of Computers Technical Report No. 143, University of Michigan.