NASA/CR-1999-209000
ICASE Interim Report No. 35

The Krigifier: A Procedure for Generating Pseudorandom Nonlinear Objective Functions for Computational Experimentation

Michael W. Trosset
The College of William and Mary, Williamsburg, Virginia

Institute for Computer Applications in Science and Engineering
NASA Langley Research Center
Hampton, VA

Operated by Universities Space Research Association

National Aeronautics and
Space Administration

Langley Research Center
Hampton, Virginia 23681-2199

Prepared for Langley Research Center
under Contract NAS1-97046

February 1999

THE KRIGIFIER: A PROCEDURE FOR GENERATING PSEUDORANDOM NONLINEAR OBJECTIVE FUNCTIONS FOR COMPUTATIONAL EXPERIMENTATION

Michael W. Trosset[1]

Abstract. Comprehensive computational experiments to assess the performance of algorithms for numerical optimization require (among other things) a practical procedure for generating pseudorandom nonlinear objective functions. We propose a procedure that is based on the convenient fiction that objective functions are realizations of stochastic processes. This report details the calculations necessary to implement our procedure for the case of certain stationary Gaussian processes and presents a specific implementation in the statistical programming language S-PLUS.

Key words. kriging, stochastic process, nonlinear programming, numerical optimization, computational experiments

Subject Classification. Applied and Numerical Mathematics

1. Introduction. It is widely accepted that the performance of algorithms for numerical optimization should be established in fact as well as in theory. Factual evidence includes the anecdotal experiences of users, but it should also include (as do other empirical sciences) the results of carefully designed experiments. Unfortunately, it is not at all clear how to design meaningful computational experiments for numerical optimization. This report attempts to address that concern.

Individuals who study numerical optimization often recommend specific algorithms for specific applications. Typically, such recommendations are based partly on theory, partly on knowledge that the recommended algorithm has performed well on other, related applications. The latter rationale implicitly assumes that the relevant population of applications has been sufficiently well sampled to warrant making predictions about the new application in question. Is this usually the case?

Computational experiments designed to assess the performance of algorithms for numerical optimization have traditionally used a small number of canonical test problems. Most of these problems were created or discovered because they exhibit some special sort of pathology. Thus, the fundamental premise of most computational experiments for numerical optimization is the following: the performance of an algorithm in typical situations can be inferred from its performance in certain pathological situations. Sadly, this premise seems dubious at best.

Consider, for example, the simplex algorithm(s) for linear programming. In theory, the computational complexity of these algorithms is exponential; in practice, they invariably perform as if their complexity was polynomial. This discrepancy between worst-case and average-case performance had led some researchers to initiate theoretical studies of *expected* simplex performance on some simple populations of randomly generated linear programs. Although realistic distributions of linear programs undoubtedly render theoretical investigations intractable, one might still study empirical simplex performance on such populations.

[1]Department of Mathematics, College of William & Mary, P.O. Box 8795, Williamsburg, VA 23187-8795 (email: `trosset@math.wm.edu`). This research was supported by the National Aeronautics and Space Administration, under NASA Contract No. NAS1-97046, while the author was in residence at the Institute for Computer Applications in Science and Engineering (ICASE), NASA Langley Research Center, Hampton, VA 23681-2199. The name "krigifier" was suggested by Robert Michael Lewis, with whom the author had many helpful conversations.

As difficult as it may be to randomly generate plausible linear programs, it seems far more difficult to randomly generate plausible nonlinear programs. This report addresses one facet of the problem of generating random nonlinear programs, viz. the problem of generating random nonlinear objective functions.

2. Basic Concepts. Originally developed by geostatisticians, kriging is a procedure for optimally interpolating a finite number of observed values of a realization of a specified stochastic process. (In case the stochastic process is an unspecified member of a specified parametric family of stochastic processes, kriging is preceded by estimation of the unspecified parameters.) The function $\hat{f}$ obtained by kriging values $y_1, \ldots, y_n$ observed at locations $x_1, \ldots, x_n$ is the expected value of the process, conditional on the process behaving as observed at $x_1, \ldots, x_n$. Thus, $\hat{f}$ can be regarded as a smoothed realization of the process and, *ceteris parabis*, the degree of smoothing depends on how many values were observed: as more and more values are kriged, $\hat{f}$ looks more and more like an actual realization.

As described in [1], the design and analysis of computer experiments is predicated on the fiction that the output from an expensive deterministic computer simulation resembles a realization of a stochastic process. We emphasize that this narrative is entirely fictional, convenient because it suggests plausible designs and analyses; nevertheless, simulations of complex physical phenomena often produce approximation, rounding, and truncation errors that contaminate the idealized output. Such deterministic noise can indeed resemble a realization of a stochastic process, so that it seems perfectly reasonable to synthesize inexpensive functions that approximate expensive simulation outputs by generating realizations of a stochastic process and adding each realization to a prescribed trend.

Conceptually, the krigifier comprises the following steps:

1. The user specifies an underlying trend, e.g. a quadratic function.
2. The user specifies a stochastic process, e.g. a stationary Gaussian process.
3. A finite number of points, $x_1, \ldots, x_n$, are chosen at which the stochastic process will be observed. These points can be specified by the user or randomly generated by the krigifier.
4. The krigifier generates $y_1, \ldots, y_n$, the values of the stochastic process at $x_1, \ldots, x_n$.
5. The krigifier interpolates $y_1, \ldots, y_n$ to obtain a noise term.
6. The trend and noise terms are added to produce an objective function.

The next section describes each of these steps.

3. Computational Details. This section describes precisely how the krigifier generates a pseudorandom nonlinear objective function; an implementation in the statistical programming language S-PLUS is provided in the following section.

1. Trend. A function trend(x) is specified by the user. This might be a constant, e.g. trend$(x) = 0$, but it seems more sensible to induce some underlying structure appropriate for nonlinear optimization, perhaps by specifyng a convex quadratic function.

2. Stochastic Process. A stochastic process is specified by the user. The process should be one from which it is reasonably easy to generate a realization. We have experimented with stationary Gaussian processes with covariance functions of the form

$$c(s,t) = \sigma^2 r(s,t), \tag{1}$$

where σ^2 is the constant variance of the process at any point (the process is homoschedastic) and the correlation function is of the form

$$r(s,t) = \phi(\|s-t\|) \tag{2}$$

(the process is isotropic). Specifically, we have experimented with

$$\phi(u) = \exp(-\theta u^\alpha) \tag{3}$$

for $\alpha = 2$ and $\alpha = 1$. The former choice results in smooth (C^∞) interpolations that seem better suited to generating "nice" objective functions; the latter choice results in jagged interpolations that seem better suited to simulating numerical noise.

3. Selected Sites. The user must specify n, the number of sites at which the stochastic process will be observed. The locations of the sites can be chosen by any method whatsoever. In our experiments, we have specified a rectangle and drawn $x_1, \ldots, x_n$ from a uniform distribution on the rectangle.

4. Observed Values. Assume that the specified stochastic process is of the form described above. Given $x_1, \ldots, x_n$, let

$$R = [r(x_i, x_j)]$$

be the $n \times n$ matrix of interpoint correlations. We need to generate $y = (y_1, \ldots, y_n)'$ by sampling from an n-variate normal distribution with covariance matrix $\sigma^2 R$. To do so, we exploit the fact that

Theorem 1 *If $z \sim N(0, I)$, then $Az \sim N(0, AA')$.*

First, we generate n standard univariate normal random variates, $z_1, \ldots, z_n$. Next, assuming that R is positive semidefinite, let $R = UDU'$ be its singular value decomposition. Then, letting $z = (z_1, \ldots, z_n)'$, Theorem 1 tells us to set

$$y = \sigma U D^{1/2} U' z.$$

5. Interpolation. We interpolate by kriging. Assuming that R is invertible, define v by the square system of linear equations $Rv = y$. The information needed to define the interpolating function is contained in $x_1, \ldots, x_n$, v, and the correlation function $r(\cdot, \cdot)$.

Given x, let

$$r(x) = \begin{bmatrix} r(x_1, x) \\ \vdots \\ r(x_n, x) \end{bmatrix}.$$

Then the interpolating function is

$$\text{noise}(x) = v' r(x).$$

6. Additive Noise. The proposed pseudorandom objective function is

$$f(x) = \text{trend}(x) + \text{noise}(x).$$

4. An Implementation in S-PLUS. This section exhibits S-PLUS functions that perform the calculations detailed in Section 3. The function `krigify`, exhibited in Figure 1, produces the information needed to define the noise term in the pseudorandom objective function f. The function `f.rand`, exhibited in Figure 2, evaluates f, which is constructed by adding the noise to a user-specified quadratic trend.

```
function(a,b,n,alpha,theta,sigma2)
{
#
#  [a,b] is a p-dimensional rectangle;
#  n is the number of sites to be selected;
#  alpha, theta, sigma2 are parameters of an isotropic
#     stationary Gaussian process.
#
   tol <- 1e-007
   p <- length(a)
   X <- matrix(runif(n*p,min=a,max=b), byrow=T, nrow=n, ncol=p)
   R <- matrix(0, nrow=n, ncol=n)
   for (i in 2:n) {
      for (j in 1:i) {
         R[i,j] <- exp(-theta * (vecnorm(X[i,]-X[j,]))^alpha)
      }
   }
   R <- R + t(R) + diag(n)
   Rsvd <- svd(R)
   d <- Rsvd$d[Rsvd$d >= tol]
   k <- length(Rsvd$d) - length(d)
   d <- c(1/sqrt(d), rep(0,times=k))
   y <- matrix(rnorm(n,sd=sqrt(sigma2)), ncol=1)
   v <- Rsvd$u %*% diag(d) %*% t(Rsvd$u) %*% y
   return(list(X=X, v=v))
}
```

Figure 1: The S-PLUS function `krigify`.

To use `krigify`, it is necessary to specify a p-dimensional rectangle $[a, b]$, the number n of sites to be selected from $[a, b]$, and the parameters $(\alpha, \theta, \sigma^2)$ of a stationary Gaussian process with an isotropic covariance function of the form specified by equations (1), (2), and (3). The sites $x_1, \ldots, x_n$ are drawn from a uniform distribution on $[a, b]$.

Once the output from `krigify` has been saved, e.g. by the S-PLUS command

```
>  noise <- krigify(a,b,n,alpha,theta,sigma2)
```

then it can be supplied to the pseudorandom objective function f whenever a function value is requested. This is accomplished by the S-PLUS function `f.rand`, exhibited in Figure 2. The function `f.rand` has two arguments, the x at which a function value $f(x)$ is requested and a list of auxiliary parameter values that specify f, and it returns $f(x)$.

```
function(x,aux)
{
#
#  x is a p-dim vector at which f is to be evaluated;
#  aux is a list:
#    aux$beta0 is a scalar,
#    aux$beta1 is a px1 matrix,
#    aux$beta2 is a pxp matrix, and
#    aux$x0 is a p-dim vector that specify the quadratic trend;
#    aux$alpha & aux$theta specify the correlation function;
#    aux$X is an nxp matrix and
#    aux$v is an nx1 matrix outputted from krigify.
#
   n <- nrow(X)
   r <- matrix(nrow=n, ncol=1)
   for (i in 1:n) {
      r[i,1] <- exp(-aux$theta * (vecnorm(X[i,1]-x))^aux$alpha)
   }
   x <- matrix(x-aux$x0, nrow=length(x), ncol=1)
   q <- aux$beta0 + t(aux$beta1) %*% x + t(x) %*% aux$beta2 %*% x
   return(q + t(aux$v) %*% r)
}
```

Figure 2: The S-PLUS function `f.rand`.

To illustrate the use of `krigify` and `f.rand`, Figure 3 exhibits S-PLUS code for generating a pseudorandom objective function f on $[0,1]^2$, evaluating f on a grid, and displaying the resulting function values in a perspective plot.

5. Conclusions. We invite the reader to experiment with the krigifier and discover parameter settings useful for his or her applications. In our view, no amount of discussion can substitute for personal experience. Nevertheless, the krigifier does exhibit certain characteristics that deserve mention.

1. Suppose that trend(x) is constant so that $f(x) = c + \text{noise}(x)$. By construction, noise$(x_i) = y_i$ and noise(x) tends to intermediate values of y for $x \notin \{x_1, \ldots, x_n\}$. Hence, the global minimizer of f in $[a,b]$ will either equal or be near the global minimizer of f in the finite set $\{x_1, \ldots, x_n\}$. Because it is generally quite difficult to construct functions with multiple local minimizers and know the location of the global minimizer, the krigifier would appear to be especially useful for constructing global optimization test functions.

```
>  a <- c(0,0)
>  b <- c(1,1)
>  noise <- krigify(a,b,200,1,50,100)
>  beta1 <- matrix(0,nrow=2,ncol=1)
>  beta2 <- 100*diag(2)
>  x0 <- c(0.3,0.4)
>  aux <- list(beta0=50, beta1=beta1, beta2=beta2, x0=x0,
              alpha=1, theta=50, X=noise$X, v=noise$v)
>  x <- y <- (0:50)/50
>  z <- matrix(nrow=51,ncol=51)
>  for (i in 1:51) {
+     for (j in 1:51) {
+        z[i,j] <- f.rand(c(x[i],y[j]), aux)
+     }
+  }
>  persp(x,y,z)
```

Figure 3: Using `krigify` and `f.rand`.

2. Our own experience with the krigifier suggests that it is easier to construct functions with multiple local minimizers in low-dimensional spaces than it is in high-dimensional spaces. To provide a heuristic explanation of this phenomenon, suppose that $\{x_1, \ldots, x_n\} \subset [a, b] \subset \Re^p$ form a rectangular grid. A local minimizer will be induced at the grid point x if each of the random variates assigned to the $2p$ grid points adjacent to x exceeds the random variate assigned to x. Obviously, the probability of this occurring decreases as p increases. To the extent that realizations of stochastic processes are indeed plausible models of objective functions, this insight suggests that functions of many variables may be *less* likely to have multiple local minimizers than functions of few variables, an amusing reversal of the curse of dimensionality.

REFERENCES

[1] J. Sacks, W.J. Welch, T.J. Mitchell, and H.P. Wynn, *Design and analysis of computer experiments*, Statistical Science **4** (1989), pp. 409–435 (includes discussion).