

# Optimal tolerance allotment using a genetic algorithm and truncated Monte Carlo simulation

J Lee and G E Johnson

---

As is typical of stochastic-optimization problems, the multivariate integration of the probability-density function is the most difficult task in the optimal allotment of tolerances. In this paper, a truncated Monte Carlo simulation and a genetic algorithm are used as analysis (i.e. multivariate-integration) and synthesis (i.e. optimization) tools, respectively. The new method has performed robustly in limited experiments, and was able to provide a significant reduction in optimal cost when compared with results published in a previous study.

**Keywords:** tolerance allotment, tolerance optimization, stochastic optimization, Monte Carlo simulation, genetic algorithms

---

The allotment of tolerances is closely tied to the overall quality and cost of a product<sup>1</sup>. If the tolerances are too loose, the probability of an assembly functioning acceptably (*yield*) is low. On the other hand, if the tolerance is too tight, the manufacturing cost becomes high. Thus tolerance allotment becomes an optimization problem to determine the optimal allotment of the tolerances under the constraints of the function requirements and acceptance probability (*spec yield*). In the design centring problem, nominal dimensions are changed to find the maximum yield with fixed tolerances, and hence the design variables are the nominal dimensions.

Methods for calculating yield (i.e. multidimensional integration) are typically classified as either *approximate methods* or *stochastic methods* (e.g. Monte Carlo simulation<sup>2</sup>). Approximate methods try to calculate the

yield by modifying the tolerance domain. In problems with low dimensions, original tolerance domains are approximated as simple regions such as spheres, cubics, or simplices<sup>3-5</sup>. Other approaches for approximating discrete distributions have also been tried<sup>6</sup>.

Because approximate methods change the domains and/or distributions, the calculated approximate yield is not the same as the yield for the original domain. As a consequence, when approximate methods are used to solve the tolerance-optimization problem, the reported yield may deviate significantly from the spec yield<sup>7</sup>. Since the spec yield is directly related to the cost, this is undesirable.

The computed yield is virtually the same as the actual yield if accurate stochastic-analysis methods (e.g. Monte Carlo simulation) are used. Unfortunately, Monte Carlo simulation (and other stochastic methods) require many sampling points to assure high accuracy. Because of the amount of computational effort required, Monte Carlo simulation may be unsuitable for the inner iteration of many classical optimization algorithms, although the rapidly decreasing cost of computing suggests that this may not be the case forever.

In the method presented here, Monte Carlo simulation is used in the tolerance-analysis procedure (i.e. yield estimation). However, instead of approximating the integration domain, the Monte Carlo simulation is approximated by truncating it after a very small number of sampling points. This greatly reduces the computational effort required for the Monte Carlo simulation. However, the resulting noise level in the objective function tends to cause problems for traditional descent methods.

Accordingly, a genetic algorithm is used in the tolerance-synthesis procedure. Genetic algorithms behave

---

Design Laboratory, University of Michigan, Ann Arbor, MI 48109-2125, USA

Paper received: 15 August 1992. Revised: 14 March 1993

robustly in erroneous and noisy environments. Results reported here show that the genetic algorithm can lead to good (i.e. cost-effective) tolerance assignment, even though the tolerance-analysis problem is only approximately solved in each function evaluation.

**STOCHASTIC-OPTIMIZATION AND TOLERANCE-OPTIMIZATION PROBLEMS**

The stochastic-optimization problem can be written as follows.

Minimize

$$E[f(\mathbf{x}, \xi)] \tag{1}$$

subject to

$$E[g_i(\mathbf{x}, \xi)] \leq 0 \quad i = 1, 2, \dots, r$$

$$\mathbf{x} \in R^n$$

$$\xi \in R^m$$

where

$$E[f(\mathbf{x}, \xi)] = \int f(\mathbf{x}, \xi) dP(\xi)$$

$$E[g_i(\mathbf{x}, \xi)] = \int g_i(\mathbf{x}, \xi) dP(\xi)$$

and  $P$  is the probability-density function of  $\xi$ .

Most of the effort required to solve this problem is spent in the multivariate integration. The domain of the integration can become unmanageable, especially when the dimensions are high and/or the constraints are complicated. In low-dimensional problems, the domains may be approximated as cubes, spheres, or simplices<sup>3-5,8,9</sup>. However, most domain-simplification schemes become impractical when the dimensionality of the problem becomes high<sup>10</sup>. A number of schemes have been developed to overcome these difficulties<sup>6,11-13</sup>.

In general, stochastic-optimization methods can be classified into three categories: common nonlinear-programming methods with simulation<sup>14,15</sup>, approximation methods<sup>6</sup>, and stochastic quasigradient methods<sup>16-18</sup>.

**Tolerance-optimization problems**

In the assembly process, some dimensions interact with other dimensions, and have effects on the function of the assembly. Those dimensions are called *sum dimensions*. For instance, consider the case of assembling a shaft and a bore, as shown in Figure 1. One of the important sum dimensions in the assembly is the clearance between the shaft and the bore.

Suppose that the gap must be at least 0.001 in, but not more than 0.005 in. In mathematical terms, then, the gap must satisfy

$$F_1(\mathbf{x}) = x_1 - x_2 - 0.001 > 0$$

$$F_2(\mathbf{x}) = -x_1 + x_2 + 0.005 > 0$$

Therefore, two constraints are generated by restricting the range of a sum dimension. In tolerancing problems, the constraints are called *design functions*. The design functions can be nonlinear if there exist angular dimensions. In practical design tasks, there exist numerous sum dimensions and their attendant constraints.

It is commonly assumed that individual dimensions follow normal distributions with a standard deviation of approximately one-sixth of the tolerance. The design functions and the acceptable tolerance regions can be projected onto the dimension space. The region that simultaneously satisfies both the design functions and the tolerances is the region in which one can expect to obtain reliable performance, and hence it is called the *reliable region*. The reliable region for the assembly in Figure 1 is shown in Figure 2.

The *yield* is computed as the probability that  $\mathbf{x}$  will fall in the reliable region. Let  $x_{iu}$  and  $x_{il}$  represent the upper and lower limits of an individual dimension  $x_i$  in an assembly. Then the yield is represented as

$$Y = \int_{x_{i1}}^{x_{i1u}} \dots \int_{x_{inl}}^{x_{inu}} q(x_1, \dots, x_n) \phi(x_1, \dots, x_n) dx_1 \dots dx_n \tag{2}$$

where  $\phi(x_1, \dots, x_n)$  is the multivariate normal probability-density function, and  $q(x_1, \dots, x_n)$  is a test function which checks whether a stochastically selected point is in the reliable region or in the infeasible region, and is defined as  $q(x_1, \dots, x_n) = 1$  if  $F_i(x_1, \dots, x_n) > 0$  for all design functions, and  $q(x_1, \dots, x_n) = 0$  otherwise.

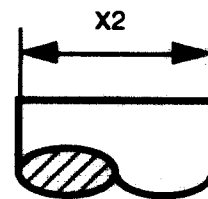
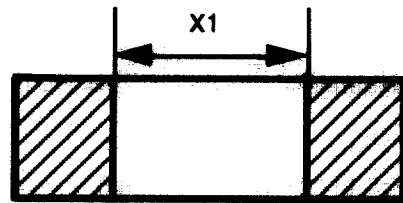


Figure 1 Assembling shaft and bore

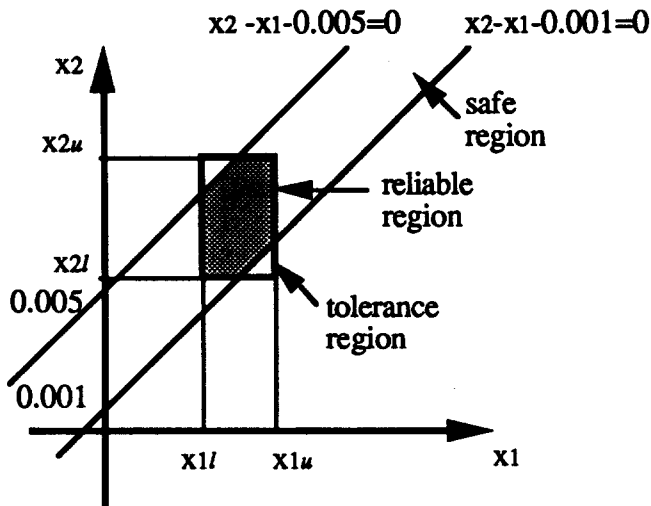


Figure 2 Reliable region as intersection of safe and tolerance region [Case of assembly in Figure 1.]

On the other hand, in a more condensed form, the yield is

$$Y = \int_{x \in R_R} \phi(x) dx \tag{3}$$

where  $R_R$  is the reliable region.

Two common cost-function models are the *reciprocal-squared model*, by Hiller<sup>19</sup>, and the *exponential model*, by Wilde and Prentice<sup>20</sup>. In the reciprocal-squared model, the cost function is represented as

$$C(t) = \frac{a}{t^2} + f \tag{4}$$

In the exponential model,

$$C(t) = a \exp\left(-\frac{t}{b}\right) + f \tag{5}$$

where  $a$  and  $b$  are constants for the variable manufacturing cost, and  $f$  is a constant for the fixed manufacturing cost.

In a multidimensional model, the total manufacturing cost can be obtained by summing the individual costs for each dimension:

$$C(\mathbf{t}) = \sum_{j=1}^n C(t_j) \tag{6}$$

or, in the standard-deviation domain,

$$C(\sigma) = \sum_{j=1}^n C(\sigma_j) \tag{7}$$

The yield increases as the tolerances become tighter. However, manufacturing costs also increase as the tolerances become tighter. The effect of individual

tolerance improvement on the total manufacturing-cost increase is different for each tolerance. Therefore, the allotment of individual tolerances which minimizes the cost under the design functions and minimum acceptable-yield constraints becomes an optimization problem. This optimal tolerance-allotment problem can be written as follows.

$$\begin{aligned} &\text{Minimize} \\ &C(\mathbf{t}) \end{aligned} \tag{8}$$

subject to

$$Pr(\mathbf{R}_R) > Y_{spec}$$

where  $Y_{spec}$  is the spec yield,  $R_R$  is the reliable region, and  $Pr(\mathbf{R}_R)$  is the estimated yield, or minimize

$$C(\sigma) \tag{9}$$

subject to

$$\int_{x \in R_R} \phi(x) dx \geq Y_{spec}$$

where  $\sigma_i \geq 0$  for  $i = 1, 2, \dots, n$ .

In optimal tolerance-allotment problem formulations,  $\sigma$  and  $\mathbf{t}$  are deterministic design variables, and the variable  $\mathbf{x}$  is stochastic. Therefore, optimal tolerance-allotment problems are stochastic constraint-optimization problems.

Lee and Woo<sup>7</sup> made progress on optimal tolerance-allotment problems by clearly defining the problem as an optimization problem using the reliability-index method, and establishing the solution steps. Their algorithm is mathematically and computationally attractive. The domains of the integration are simplified into a half space or into a hypersphere. However, their approximation schemes do not give solutions to problems where there is an initially specified minimum acceptable yield (*spec yield*). The main reason for this is that the real yield for the original domain is not calculated in their algorithm. The real yields are calculated by a Monte Carlo method after the problem has been solved. To overcome this, they assumed that reliability would be equally distributed on each constraint. The results of Lee and Woo are frequently cited in subsequent sections, and the results of the new methods are compared with theirs where appropriate.

### Integration of genetic algorithm and Monte Carlo simulation

The outline of the proposed procedure is shown in *Figure 3*. At the first step, a set of variables is selected as the initial population. Then, the cost function is evaluated by Monte Carlo simulation in the analysis step. As a synthesis step, a genetic algorithm<sup>21</sup> perturbs the

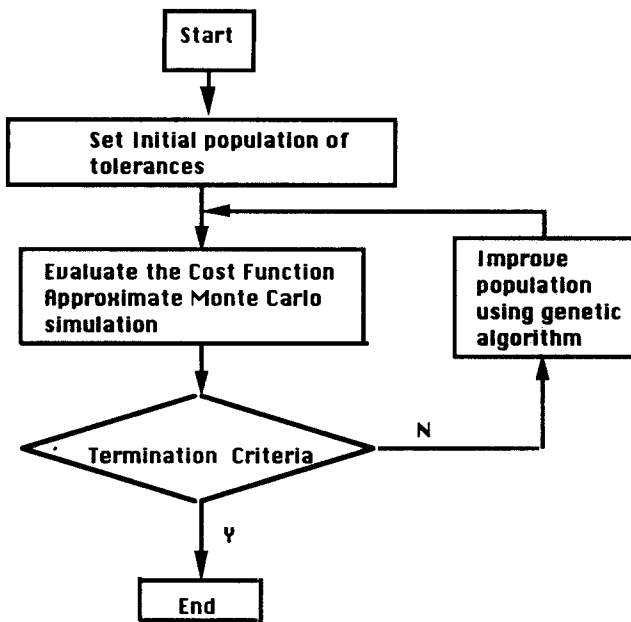


Figure 3 Flowchart of new algorithm

population using the three steps of reproduction, crossover and mutation. The algorithm iterates until the termination criteria are satisfied. Various termination criteria can be used. In this research, sufficiently large generation numbers are set as termination generations to investigate the behaviour of the algorithm.

**Cost-function formulation using penalty-function method**

A simple way of dealing with the constraints is the penalty-function method<sup>22,23</sup>. One common penalty function is the quadratic exterior penalty function, which assigns a penalty that is proportional to the sum of the squares of the constraint violations. The optimal tolerance-allotment problem (see Expression 9) can now be replaced by the following.

Minimize

$$\Psi(\sigma, r) \tag{10}$$

where

$$\Psi = C(\sigma) + r \langle -g \rangle^2$$

$$g = \int_{x \in R_x} \phi(x) dx - Y_{spec}$$

where

$$\langle a \rangle = \begin{cases} a & a \geq 0 \\ 0 & a < 0 \end{cases}$$

and  $r$  is a penalty coefficient and positive, or minimize

$$C(\sigma) + r \left\langle - \left( \int_{x \in R_x} \phi(x) dx - Y_{spec} \right) \right\rangle^2 \tag{11}$$

The minimum of the exterior penalty-function problem is theoretically in the infeasible region with a slight constraint violation for finite values of  $r$ . However, with the genetic algorithm, the candidate solution points are distributed around the minimum. Therefore, there is a finite probability that there are solutions in the feasible region.

**Coding of strings**

The population structure of tolerance allotment is

$$\begin{aligned} \sigma_1 &= (\sigma_{11}, \sigma_{12}, \dots, \sigma_{1n}) \\ \sigma_2 &= (\sigma_{21}, \sigma_{22}, \dots, \sigma_{2n}) \\ &\vdots \\ \sigma_p &= (\sigma_{p1}, \sigma_{p2}, \dots, \sigma_{pn}) \end{aligned} \tag{12}$$

where  $p$  is the size of the population.

The strings represent discretized points in the continuous domain. The continuous design domain should be transformed to the discrete design domain using a discretization and coding process. DeJong<sup>24</sup> discusses a mapped, fixed-point parameter where a  $j$  bit substring is interpreted as the usual, unsigned binary integer on the interval  $[0, 2^j - 1]$ . This integer is linearly mapped onto some specified interval of the real numbers. In tolerance-allotment problems, a string is mapped onto 0 to  $3\sigma$  in the standard-deviation domain. Therefore, the precision of coding  $\pi$  is given by

$$\pi = \frac{3\sigma}{2^j - 1} \tag{13}$$

In tolerance-allotment problems, a chromosome is a combination of parameter substrings which represent each dimension's standard deviation. For the linear tolerance-allotment problem described below, there exist eight design variables, and a substring is composed of 6 bit. Therefore, the length of a chromosome is 48 (i.e. 6 multiplied by 8). A substring is interpreted as an integer on the interval  $[0, 63]$ . Then, the decoded integer is linearly mapped onto the real numbers  $[0, 3\sigma_{max}]$  in the standard-deviation domain. Figure 4 shows the decoding and mapping procedure.

The precision  $\pi$  of this coding is  $0.0471\sigma_{max} (= 3\sigma_{max}/63)$ . The mapping equation is

$$\sigma = (\text{decoded integer})(0.0471\sigma_{max})$$

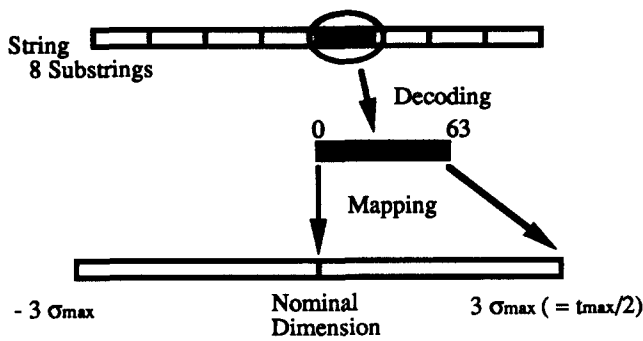


Figure 4 Decoding and mapping

**Fitness inversion**

To maximize the fitness, the strings identified as 'most fit' receive more chances to produce children than the strings of low fitness in each reproduction step. However, in the tolerance-allotment problem, the objective is the minimization of the cost. Accordingly, we reciprocate our objective function for minimization and maximize, i.e.

$$\hat{f}(x) = \frac{c}{f(x)} \tag{14}$$

where  $f(x)$  is an original fitness function, and  $c$  is a positive constant.

**Modified linear-fitness scaling**

The original fitness proportionate-reproduction scheme frequently causes two significant difficulties: premature termination at early generations, and stalling at late generations. At the early stage, there exist some singular strings which have excessively large fitness values in comparison with the fitness values of other strings. Therefore, these strings dominate reproduction. As a consequence, the string pattern in the population shows similarities with these dominating strings after several generations. This reduces the chances of searching diverse string combinations, and could result in premature termination at a suboptimal point. In the late stages, the highest fitness of a string may be slightly higher than the average population fitness, even though there exist diversities between the strings. In this situation, there is no improvement toward the optimum. Therefore, it is necessary to modify the original fitness proportionate-reproduction schemes. Goldberg<sup>25</sup> suggested a *linear-fitness scaling* scheme. The relation between the original fitness  $u$  and the linearized fitness  $u_s$  is

$$u_s = au + b$$

Our experience suggests that a modified linear-fitness scaling scheme where the strings in a population are divided into two groups (lower-fitness strings and higher-fitness strings) by comparison of their fitnesses

with the average fitness of the population may be superior for some problems. For the lower-fitness strings, the minimum original fitness  $u_{min}$  maps to 0. For the higher-fitness strings, the maximum original fitness  $u_{max}$  maps to  $f_{multiple}$ . The average fitness  $u_{avg}$  maps to 1. The new mapping scheme is shown in Figure 5.

**Parameters**

For the tolerancing problems, the parameters are selected as follows:

- population size = 100
- crossover probability = 0.7
- mutation probability = 0.005

**Experiments**

The computational experiments were performed on an Apollo DN 5500 workstation. The main purpose of the experiments was to explore whether or not the genetic algorithm would be sufficiently robust to overcome the noise of the truncated Monte Carlo simulation. At the outset, it was assumed that the genetic algorithm would require more generations to reach a certain level of performance if the function evaluations were more noisy. However, the computation time for each iteration is reduced by using the truncation strategy. Therefore, the total computation time taken to reach a solution may be shorter when a truncated (coarse) Monte Carlo simulation is used.

The performance of the Monte Carlo simulation was estimated by evaluating an 8D probability-density function with four design functions. The nominal dimensions are given as  $x^T = (1.0 \ 2.0 \ 3.0 \ 4.0 \ 1.0 \ 0.998 \ 2.0 \ 2.998)$ . Tolerances are given as  $t^T = (0.0040 \ 0.0023 \ 0.0025 \ 0.0053 \ 0.0143 \ 0.0021 \ 0.0015 \ 0.0020)$ . The design functions

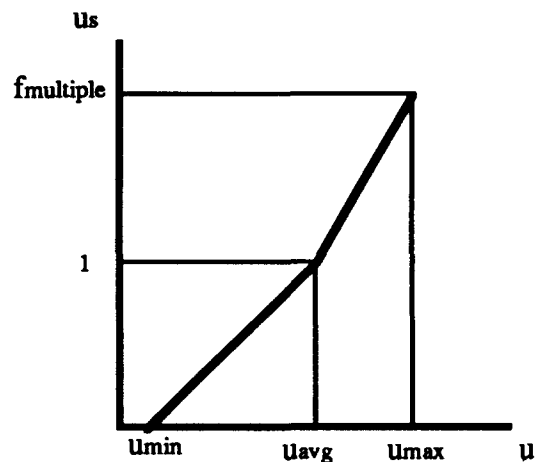


Figure 5 Modified linear-fitness scaling

**Table 1** Results of Monte Carlo simulation

Sampling numbers	Yield	Standard deviation
10	0.87089	0.33697
20	0.83625	0.21513
50	0.83724	0.18588
100	0.83229	0.15487
300	0.83460	0.06328
1000	0.83477	0.05227
3000	0.83672	0.01556
10000	0.83892	0.01170
30000	0.84067	0.00870

are as follows:

$$F_1(\mathbf{x}) = -x_4 - x_5 + 5.005$$

$$F_2(\mathbf{x}) = x_2 - x_1 - x_8 + x_7 - 0.0003$$

$$F_3(\mathbf{x}) = x_7 - x_6 - x_3 + x_2 + 0.001$$

$$F_4(\mathbf{x}) = x_4 - x_3 - x_6 - 0.0003$$

Several tests were performed for various sampling numbers. The test runs were performed ten times for each sampling number. Then the results of ten test runs were averaged, and standard deviations were calculated. The results of the simulations are shown in *Table 1*. The results show that the standard variation of the simulated yields is inversely proportional to the square root of the sample numbers. This result indicates how many sampling points are necessary to guarantee a certain level of precision. If an optimization algorithm needs one standard deviation for the simulated yield to be less than 0.01 in order to converge to a solution, then the number of sample points for Monte Carlo simulation should be more than 30 000.

This is because of the nature of the integrand. The probability-density function becomes very highly peaked, especially when the standard deviation is small. High dimensionality makes the high peakedness more severe.

*Table 1* shows that, for 100 sampling points, the precision of the estimation was found to be 0.15 for one standard deviation. The algorithm identified a good solution even for this noisy environment. The number of sampling points was gradually decreased until the algorithm could no longer find a good solution. As the next step, various sampling numbers were tried to explore the relations between the computation efforts, the approximation, and the performance.

To define the standard performance, the threshold levels for the objective function value were set slightly higher than optimum. The algorithm was run with various parameter settings. The algorithm was judged to have obtained a good solution when the minimum costs of several consecutive generations passed the threshold level. The relation between the computation time and the number of sampling points was studied after each experimental run.

## RESULTS AND ANALYSIS

### Linear-constraint problem

For the first example, a problem from Reference 7 was selected. The shape of the assembly is shown in *Figure 6*. The linear design functions for this example are

$$F_1(\mathbf{x}) = -x_4 - x_5 + 5.005$$

$$F_2(\mathbf{x}) = x_2 - x_1 - x_8 + x_7 - 0.0003$$

$$F_3(\mathbf{x}) = x_7 - x_6 - x_3 + x_2 + 0.001$$

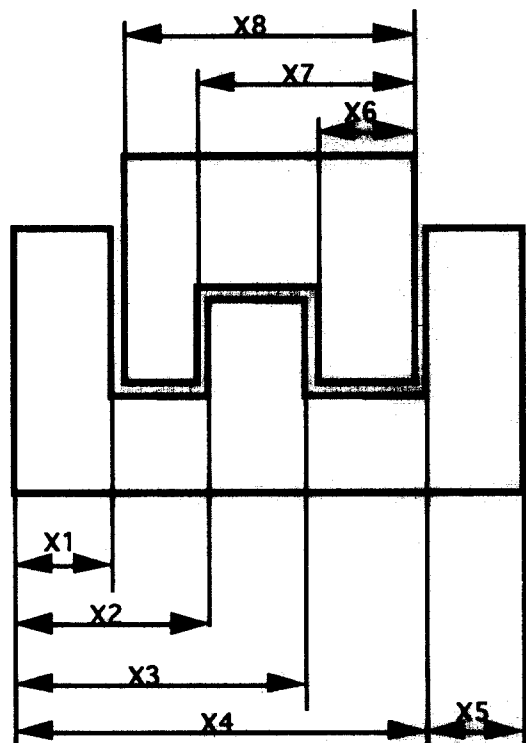
$$F_4(\mathbf{x}) = x_4 - x_3 - x_6 - 0.0003$$

$F_1(\mathbf{x})$  is the constraint on the size of the base part. The other design functions represent the clearance condition for assembly. The nominal dimensions are given as  $\mathbf{x}^T = (1.0 \ 2.0 \ 3.0 \ 4.0 \ 1.0 \ 0.998 \ 2.0 \ 2.998)$ . The reciprocal cost-function model (see Equation 4) was modified, and used to define the total manufacturing cost as

$$C_i(\sigma_i) = \frac{a_i \times 10^{-3}}{(6\sigma_i)^{b_i}} \tag{15}$$

The coefficients in Equation 15 were set by Lee and Woo as  $a_1 = a_2 = 1.0$ ,  $a_3 = a_4 = 1.5$ ,  $a_5 = 0.8$ ,  $a_6 = 0.9$ ,  $a_7 = 0.8$  and  $a_8 = 0.6$ , and  $b_1 = 2.0$ ,  $b_2 = 1.8$ ,  $b_3 = 1.7$ ,  $b_4 = 2.0$ ,  $b_5 = 3.0$ ,  $b_6 = 2.0$  and  $b_7 = b_8 = 1.9$ . The spec yield is 95%.

For the genetic algorithm, the population size is 100. Test runs were performed for various numbers of sampling points: 10, 20, 50 and 100. At the last generation, Monte Carlo simulations were performed using 10 000

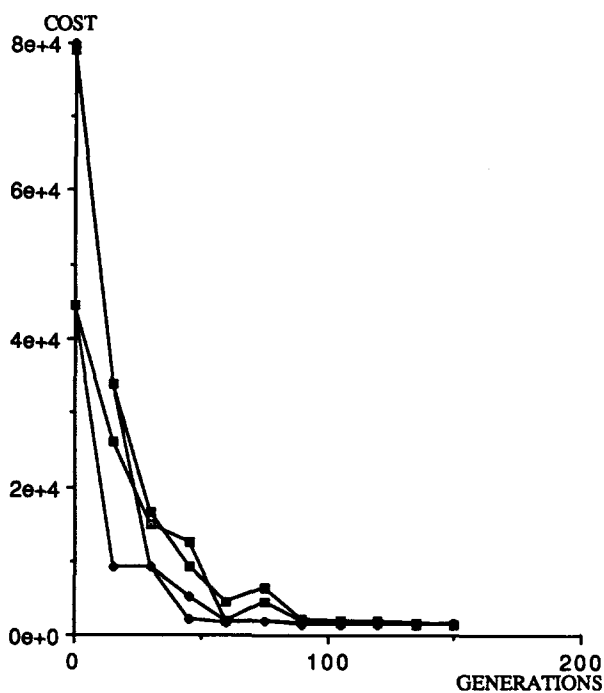


**Figure 6** Linear-constraint problem

**Table 2** Results of linear problem

Variables	Tolerances			W J Lee algorithm
	Genetic algorithm			
	String 1	String 2	String 3	
$t_1$	0.00333	0.00333	0.00333	0.00328
$t_2$	0.00133	0.00133	0.00133	0.00124
$t_3$	0.00143	0.00086	0.00086	0.00174
$t_4$	0.00305	0.00381	0.00305	0.00403
$t_5$	0.01429	0.01333	0.01333	0.01281
$t_6$	0.00171	0.00171	0.00171	0.00123
$t_7$	0.00133	0.00133	0.00133	0.00104
$t_8$	0.00143	0.00143	0.00143	0.00273
Cost	1475.84*	1618.42	1676.56	1816.38
Yield, %	94.63	95.29	96.15	94.95 <sup>†</sup>
CPU run time, s		80.2 <sup>‡</sup>		0.14 <sup>§</sup>

[\* Penalized cost: 1516.96. <sup>†</sup> 2000 run. <sup>‡</sup> Apollo DN 5500/10 sampling points. <sup>§</sup> IBM 3090-400 V/M.]



**Figure 7** Progress of algorithm  
[□: 10 samples, ◆: 20 samples, ■: 50 samples, ◇: 100 samples.]

sampling points for all strings to estimate the yields precisely. The threshold level was set at the cost of 1600. The results of a typical test run are shown in *Table 2*. For the test run, the number of sampling points was 30, and the generation number was 150.

The solutions were compared with the solution reported by Lee and Woo<sup>7</sup>. The last two rows of *Table 2* show the costs and yields. The results clearly show that the new algorithm gives better solutions, because the costs were significantly reduced (by about 12%) for similar yields. Of course, since theoretical evaluation of the method proposed here is not available at this time, there is no guarantee that better performance will always be

obtained. However, the results are promising, are worth reporting, and warrant further investigation.

*Figure 7* shows the progress obtained by the algorithm for various sampling-point numbers. At the initial stage, the costs are extremely large, owing to the random selection of the candidate tolerances. As the generation proceeds, the costs are reduced to the optimal cost. As expected, the algorithm obtains a good solution in fewer generations when larger sampling numbers are used for the Monte Carlo simulation.

The computational complexity of an algorithm can be represented by various measurements, such as the number of arithmetic operations, the number of function evaluations, the number of iterations, and the computation time. The *time complexity* of the computation in this algorithm is the sum of the time spent for the Monte Carlo simulation and the time spent for the genetic algorithm. Simple experiments were performed to compare the computation time for the Monte Carlo simulation and the genetic algorithm. In performing the algorithm with 100 sampling points and 100 generations, the CPU time spent for the Monte Carlo simulation and the genetic algorithm was 791.6 s and 10.8 s, respectively, on an Apollo DN 5500 workstation. Therefore, the computation time for the genetic algorithm is very small in comparison with the time associated with the Monte Carlo simulation. Because the computation time for the Monte Carlo simulation is exactly proportional to the number of the sampling points, the *computation effort* is defined as the generation number multiplied by the sampling-points number, and this product can be used as a measurement of the time complexity. In solving this problem, an average of 171 were required for 10 sampling points. Therefore, the computation complexity is 1710 in this case.

*Figure 8* shows the progress of the algorithm in terms of the computation effort. It shows that the algorithm progresses faster when a smaller number of sampling

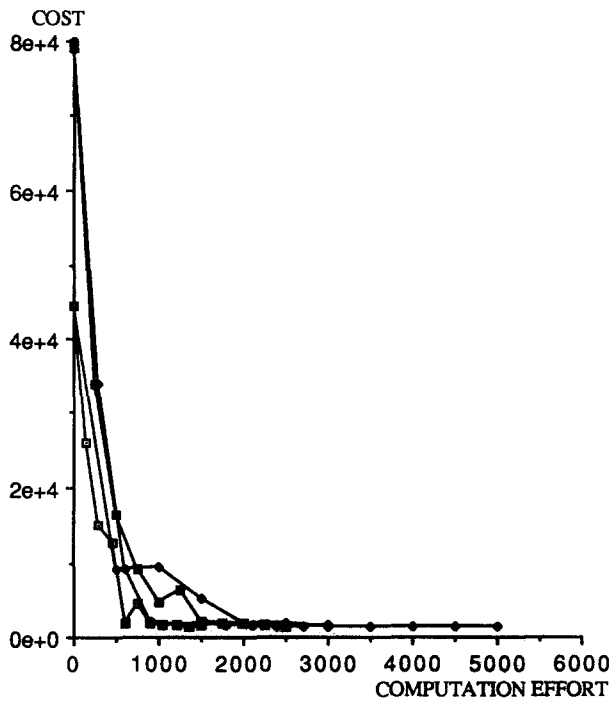


Figure 8 Progress with respect to computation effort [□: 10 samples, ◆: 20 samples, ■: 50 samples, ◇: 100 samples.]

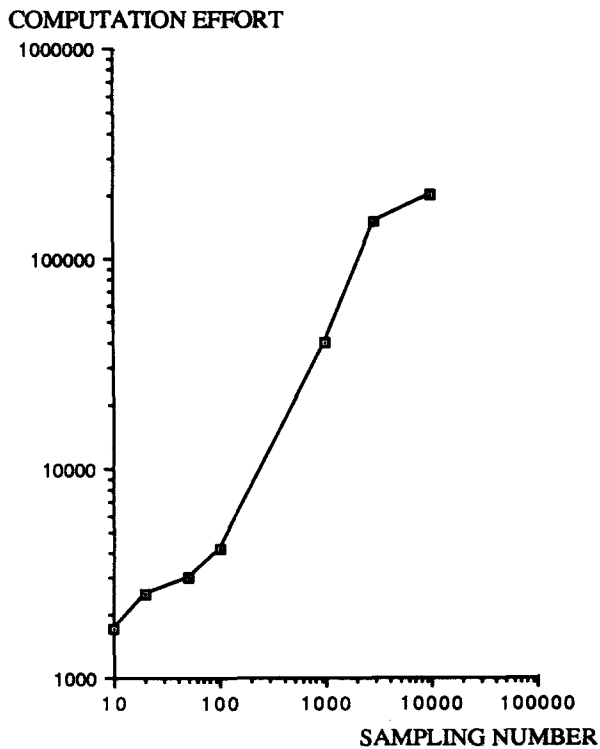


Figure 9 Sampling numbers against computation effort

points are used. The computation effort to reach the threshold level is shown in Figure 9.

To examine the relation between the computation effort and the point of truncation for the simulation, seven different sampling numbers, 10, 20, 50, 100, 1000, 3000 and 10000 sampling points, were used for the Monte Carlo simulation. The generation numbers required to

reach the threshold level are the averaged values of four runs for each sampling number. Figure 9 shows clearly that the algorithm progresses faster (i.e. with less computation effort) when a smaller number of sampling points are used (up to the point where there are too few sampling points to ensure a good solution). The algorithm progressed to a good solution when as few as eight sampling points were used in this example.

**Nonlinear-constraint problem**

Next, a nonlinear-constraint problem posed by Lee<sup>3</sup> was tried. The shape of the assembly is shown in Figure 10, and the corresponding design functions are as follows:

$$F_1(\mathbf{x}) = (x_6 - x_5) - (x_8 - x_7)$$

$$F_2(\mathbf{x}) = (x_3 - x_4) - (x_{11} - x_{10})$$

$$F_3(\mathbf{x}) = (x_8 - x_7) * (x_2 - x_3) - (x_6 - x_5) * (x_{10} - x_9) + \tan(\pi/180) * \{(x_{10} - x_9) * (x_2 - x_3) - (x_8 - x_7) * (x_6 - x_5)\}$$

$$F_4(\mathbf{x}) = (x_6 - x_5) * (x_{10} - x_9) - (x_8 - x_7) * (x_2 - x_3) + \tan(\pi/180) * \{(x_{10} - x_9) * (x_2 - x_3) - (x_8 - x_7) * (x_6 - x_5)\}$$

$$F_5(\mathbf{x}) = -x_1 + x_{12} + 0.01$$

$$F_6(\mathbf{x}) = x_1 - x_{12} + 0.01$$

The first two design functions are the vertical and the horizontal clearance conditions of the two parts. The third and the fourth design functions restrict the difference between the angles  $\theta_1$  and  $\theta_2$  to be within  $\pm \pi/180$  rad ( $1^\circ$ ) for successful assembly. The last two conditions require the size difference between two parts to be within  $\pm 0.01$ . The nominal dimensions are given as  $\mathbf{x}^T = (50.0, 40.00125, 20.05, 9.9985, 9.9985, 30.0, 10.0, 30.0, 10.05, 30.0, 40.0, 50.0)$ . The modified cost function of Equation 15 is used.

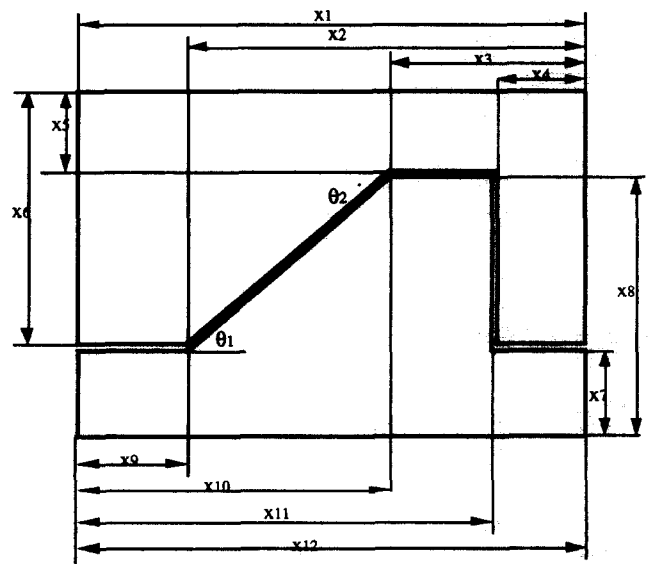


Figure 10 Nonlinear-constraint problem



**Table 3** Results of nonlinear problem

Variables	Tolerances			W J Lee algorithm
	Genetic algorithm			
	String 1	String 2	String 3	
$t_1$	0.01647	0.01647	0.01294	0.0187
$t_2$	0.18824	0.20707	0.86591	1.2331
$t_3$	0.05897	0.05189	0.03585	0.0579
$t_4$	0.05646	0.05646	0.06587	0.0705
$t_5$	0.00235	0.00235	0.00235	0.0019
$t_6$	0.00212	0.00212	0.00212	0.0022
$t_7$	0.00306	0.00306	0.00306	0.0019
$t_8$	0.00212	0.00212	0.00212	0.0015
$t_9$	0.82765	0.63847	0.82765	1.2385
$t_{10}$	0.05788	0.06212	0.05647	0.0714
$t_{11}$	0.02353	0.01224	0.02259	0.0579
$t_{12}$	0.01176	0.01176	0.01294	0.0168
Cost	7.90	7.97	8.08	10.38
Yield, %	94.91	95.40	95.47	94.50*
CPU run time, s		364.2 †		0.418‡

[\* 2000 runs. † Apollo DN 5500/30 sampling points. ‡ IBM 3090-400 V/M.]

The coefficients in Equation 15 were set by Lee as  $a_1=0.2$ ,  $a_2=1.0$ ,  $a_3=a_4=0.015$ ,  $a_5=0.008$ ,  $a_6=0.009$ ,  $a_7=0.008$ ,  $a_8=0.006$ ,  $a_9=1.0$ ,  $a_{10}=0.01$ ,  $a_{11}=0.015$  and  $a_{12}=0.2$ , and  $b_1=b_2=\dots=b_{12}=2.0$ .

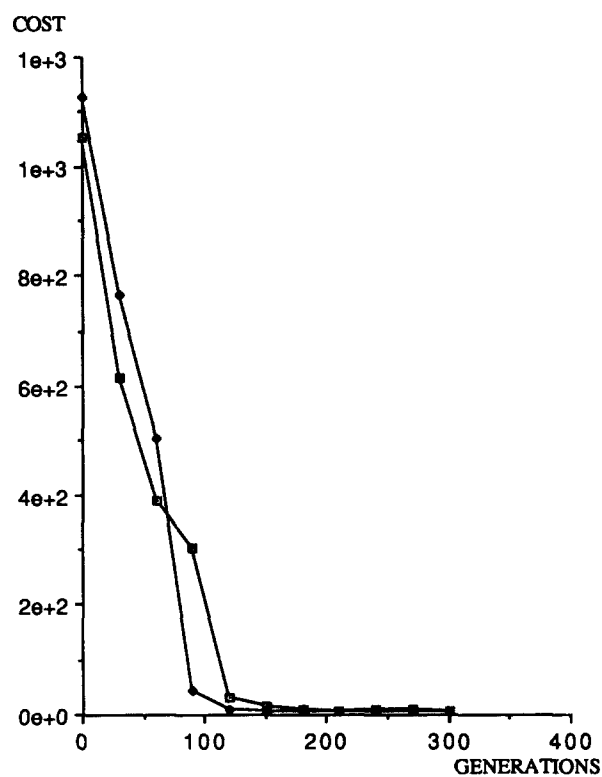
There are 12 design variables, and a substring for a variable is composed of 8 bit. Therefore, the length of a string is 96. A substring is interpreted as an integer on the interval  $[0, 255]$ . The precision  $\pi$  of the coding is  $0.011765\sigma_{\max}$ .

The results of a test run are shown in Table 3. For the test run, the number of generations was 300, and the number of sampling points was 30. The results show that the cost was reduced by about 20% in comparison with the cost obtained by Lee's algorithm<sup>3</sup>, and the yield is slightly improved. Again, we cannot guarantee that this will always be the case. However, the results are sufficiently interesting to report them here in order to spur further investigation.

To examine the relation between the computation effort and the approximation of simulation, several sampling numbers were used for the Monte Carlo simulation. Figure 11 shows the progress of the algorithm for 30 sampling points and 100 sampling points. Figure 12 shows the progress of the algorithm in terms of the computation effort for each case. Figures 11 and 12 show that the algorithm progresses faster when a smaller number of sampling points are used, even though it takes more generations.

## SUMMARY

The optimal costs were significantly reduced in comparison with the costs of the domain-approximation schemes for the cases that we have investigated. This suggests that it



**Figure 11** Progress with respect to generations  
[□: 30 samples, ◆: 100 samples.]

would be worthwhile to develop the new algorithm further, study its theoretical performance, and test it on a wider range of practical examples to see whether or not the excellent results obtained in this study will be generally duplicated.

The two problems considered here were successfully solved with *extremely small* numbers of Monte Carlo

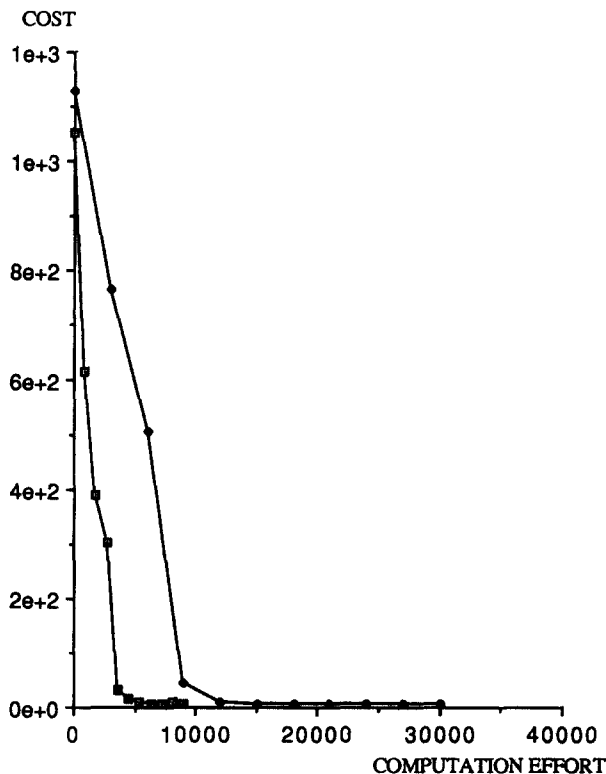


Figure 12 Progress with respect to computation effort  
[□: 30 samples, ◆: 100 samples.]

simulation sampling points. For the linear-constraint problem, the algorithm identified a good solution as long as more than eight sampling points were used. For the nonlinear-constraint problem, the algorithm successfully identified a good solution for 30 points. This might be because of the longer string structure of the nonlinear-constraint problem. Despite the coarse resolution and low precision, the algorithm satisfied the spec yield with sufficient precision. It is not possible to draw general conclusions about how many sampling points will be required in other cases. More experimental investigation could develop rules of thumb based on case studies. Theoretical investigation of the algorithm's characteristics would be the most satisfying method to address these questions.

For many mathematical-programming algorithms, the time complexity can increase geometrically with the number of variables. However, in the code developed in this paper, the time complexity increases linearly as the dimension increases. Computational experiments bear this out. Therefore, we believe that the new algorithm could offer significant advantages for high-dimension problems.

The strategy developed in this paper is unique and practical. Tolerance-optimization problems are not trivial. The cost-function modelling and the determination of the design functions takes substantial time and effort. Domain-approximation schemes offer advantages in short computation time; however, these advantages are

offset in cases in which it is important to identify the best possible solution, or in which the spec yield is constrained.

This basic idea presented here (truncated Monte Carlo simulation coupled with a genetic algorithm) could be used to solve other stochastic-optimization problems. This is a worthwhile subject for future research. A more complete description of the work reported here, along with an extensive bibliography, is given in Reference 26.

## ACKNOWLEDGEMENTS

The authors gratefully acknowledge the financial support provided by Daewoo Electronic Corporation, Seoul, Korea. The paper is based on the PhD dissertation of J Lee, which was completed under the direction of G E Johnson at the University of Michigan, USA, in August 1992.

## REFERENCES

- 1 Bjørke, O *Computer-Aided Tolerancing* ASME Press, USA (1989) pp 86–87
- 2 Hammersley, J M and Handscomb, D C *Monte Carlo Methods* Methuen, UK (1964)
- 3 Lee, W J 'Tolerancing: computations on geometric uncertainties' *Doctoral Dissertation* University of Michigan, USA (1989)
- 4 Good, I J and Gaskins, R A 'The centroid method of numerical integration' *Numer. Math.* Vol 16 (1971) pp 343–359
- 5 Director, S W and Hachtel, G D 'The simplicial approximation approach to design centering' *IEEE Trans. Circuits & Syst.* Vol CAS-24 No 7 (1977) pp 363–372
- 6 Wets, R 'Stochastic programming: solution techniques and approximation schemes' in Bachem, A, Groetschel, M and Korte, B (Eds.) *Mathematical Programming – The State-of-the-Art* Springer-Verlag, Germany (1983) pp 566–603
- 7 Lee, W J and Woo, T C 'Tolerances: their analysis and synthesis' *J. Eng. Indust.* Vol 112 (May 1990) pp 113–121
- 8 Lebedev, V I 'On cubature formulas of spheres' *J. Computat. Math. & Math. Phys.* Vol 16 (1976) pp 293–306
- 9 Freeden, W 'On integral formulas of the (unit) sphere and their application to numerical computation of integrals' *Computing* Vol 25 (1980) pp 131–146
- 10 Deak, I 'Multidimensional integration and stochastic programming' in *Numerical Techniques for Stochastic Optimization* Springer-Verlag, USA (1988) pp 187–200
- 11 Donnelly, T G 'Bivariate normal distribution' *Commun. ACM* Vol 16 (1973) p 638
- 12 Brown, J L 'On the expansion of the bivariate Gaussian probability density using results of nonlinear theory' *IEEE Trans. Inf. Theor.* (1968) pp 158, 159
- 13 Drezner, Z 'Computation of the bivariate normal integral' *Math. Comput.* Vol 32 (1978) pp 277–279
- 14 Polak, E *Computational Methods in Optimization* Academic Press, USA (1971)
- 15 Bazzaraa, M S and Shetty, C M *Nonlinear Programming: Theory and Algorithms* John Wiley, USA (1979)
- 16 Ruzynski, A and Syski, W 'Stochastic approximation algorithm with gradient averaging for unconstrained problems' *IEEE Trans. Automatic Control* Vol AC-28 (1983) pp 1097–1105
- 17 Kiwiel, K 'An aggregate subgradient method for nonsmooth convex minimization' *Math. Prog.* Vol 27 (1983) pp 320–341
- 18 Gaivoronskiy, A A 'Methods of stochastic nonstationary optimization' in *Operations Research and System Reliability* Press of the Institute of Cybernetics, Kiev, Ukraine (1978)
- 19 Hiller, M J 'A systematic approach to the cost optimization of tolerances in complex assemblies' *Bull. Mech. Eng. Educ.* Vol 5 (1966) pp 157–161
- 20 Wilde, D and Prentice, E 'Minimum exponential cost allocation

of sure-fit tolerances' *ASME Paper 75-DET-93* ASME, USA (1975)

21 Holland, J H 'Adaptation in natural and artificial systems' University of Michigan Press, USA (1975)

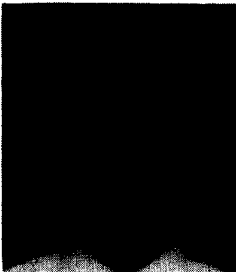
22 Carroll, C W 'The created response surface technique for optimizing nonlinear restrained systems' *Oper. Res.* Vol 9 (1961) pp 169-184

23 Fiacco, A V and McCormick, G P *Nonlinear Programming: Sequential Unconstrained Minimization Techniques* John Wiley, USA (1968)


24 DeJong, K A 'An analysis of the behavior of the class of genetic adaptive systems' *Doctoral Dissertation* University of Michigan, USA (1975)

25 Goldberg, D E *Genetic Algorithms in Search, Optimization, and Machine Learning* Addison-Wesley, USA (1989)

26 Lee, J 'Tolerance optimization using genetic algorithm and approximated simulation' *Doctoral Dissertation* University of Michigan, USA (1992)



*Jinkoo Lee received a BSME from Yonsei University, and an MSME and a PhD from the University of Michigan, USA. His research interests were in design optimization and mechanism design. He now works at Daewoo Electronics, Korea, where he designs VCR systems and studies magnetic-recording mechanisms.*



*Glen E Johnson received a BSME from Worcester Polytechnic Institute, USA, and an MSME from Georgia Tech, USA, and a PhD from Vanderbilt University, USA. He is the director of the Design Laboratory at the University of Michigan, USA, and the Chair of the American Society of Mechanical Engineers' Design Engineering Division. His research and teaching activities are in the areas of optimal design, the design of machine components, and the design of mechanical systems.*