

MARTIN MARIETTA ENERGY SYSTEMS LIBRARIES



ORNL/TM-11566

3 4456 0315029 3

# ornl

**OAK RIDGE  
NATIONAL  
LABORATORY**

**MARTIN MARIETTA**

## Piecewise Linear Models of Processor Utilization

Michael D. Vose

OAK RIDGE NATIONAL LABORATORY  
 CENTRAL RESEARCH LIBRARY  
 CIRCULATION SECTION  
 ROOM 1008 175  
**LIBRARY LOAN COPY**  
 DO NOT TRANSFER TO ANOTHER PERSON  
 If you wish someone else to see this  
 report, send in name with report and  
 the library will arrange a loan.

OPERATED BY  
MARTIN MARIETTA ENERGY SYSTEMS, INC.  
FOR THE UNITED STATES  
DEPARTMENT OF ENERGY

This report has been reproduced directly from the best available copy.

Available to DOE and DOE contractors from the Office of Scientific and Technical Information, P.O. Box 62, Oak Ridge, TN 37831; prices available from (615) 576-8401, FTS 626-8401.

Available to the public from the National Technical Information Service, U.S. Department of Commerce, 5285 Port Royal Rd., Springfield, VA 22161.

NTIS price codes—Printed Copy: A03 Microfiche A01

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

Engineering Physics and Mathematics Division  
Mathematical Sciences Section

**PIECEWISE LINEAR MODELS OF PROCESSOR UTILIZATION**

Michael D. Vose

The University of Tennessee  
Department of Computer Science  
107 Ayres Hall  
Knoxville, TN 37996

Date Published: June, 1990

Research was supported by the  
Applied Mathematical Sciences Research Program  
of the Office of Energy Research,  
U.S. Department of Energy.

Prepared by the  
Oak Ridge National Laboratory  
Oak Ridge, Tennessee 37831  
operated by  
Martin Marietta Energy Systems, Inc.  
for the  
U.S. DEPARTMENT OF ENERGY  
under Contract No. DE-AC-05-84OR21400

MARTIN MARIETTA ENERGY SYSTEMS LIBRARIES



3 4456 0315029 3



## Contents

<b>1 Introduction</b>	<b>1</b>
<b>2 Unconstrained Placement</b>	<b>1</b>
2.1 Partial Derivatives . . . . .	2
2.2 Algorithm . . . . .	4
<b>3 Constrained Placement</b>	<b>6</b>
3.1 Construction . . . . .	7
3.2 Algorithm . . . . .	9
<b>4 Applications</b>	<b>9</b>
<b>References</b>	<b>13</b>



# PIECEWISE LINEAR MODELS OF PROCESSOR UTILIZATION

Michael D. Vose

## Abstract

This paper is part of a larger research effort to characterize the performance of parallel programs on distributed memory multiprocessors. We consider modeling processor utilization data by continuous piecewise linear approximations. We describe interactive tools for the identification of underlying trends and for the suppression of superfluous detail in processor utilization graphs.



## 1. Introduction

As part of a larger research effort to characterize the performance of parallel programs on distributed memory multiprocessors, we consider estimating processor utilization data by continuous piecewise linear approximations. We eventually hope to automate the identification of underlying trends in processor utilization curves and to suppress superfluous detail. This would become an important part of the automatic generation of scalable performance models for parallel programs.

Standard data modeling and statistical techniques do not seem appropriate for this goal since abrupt changes in program behavior frequently signal distinct execution phases which should be kept by the model rather than smoothed away. Moreover, since area under a utilization curve corresponds to work, it should be preserved to the extent possible. As a first step toward realization of our goal, we describe a basic tool which allows interactive determination of the features of a utilization curve which are to be preserved by its approximation, and describe an approximation technique particularly suited for simplifying utilization data while preserving salient detail.

This paper is organized in three parts. In §2 and §3 we describe our approximation methods as abstract problems and present their analysis in general mathematical terms. The reader primarily interested in the motivation for and use of our algorithms will probably wish to skim through these sections. In §4 we illustrate the use of our techniques and discuss their suitability for our ultimate goal of automatically identifying underlying trends in processor utilization.

## 2. Unconstrained Placement

The general problem considered in this section is global approximation. The intent is that the complexity of approximation be controllable, and that overall behavior be represented while at the same time the possibility of local bias is provided for.

Let  $f$  be a function defined over the real interval  $[a, b]$ . Let  $K = \{(x_0, y_0), \dots, (x_n, y_n)\}$  be a set of points from  $\mathcal{R}^2$  satisfying

$$a = x_0 < \dots < x_n = b,$$

$$y_0 = f(a), \quad y_n = f(b).$$

Let  $l_K$  be the continuous piecewise linear function defined for  $j = 1 \dots n$  over the intervals  $[x_{j-1}, x_j]$  by

$$l_K(x) = y_{j-1} + \frac{x - x_{j-1}}{x_j - x_{j-1}}(y_j - y_{j-1}).$$

The problem we consider is: Given  $f$  and  $n$ , how may the set  $K$  be determined to minimize

$$\int_a^b (f(x) - l_K(x))^2 dx.$$

In general, this problem has no unique solution, the techniques of calculus do not yield equations which may be solved in closed form, and the surface corresponding to this minimization problem has local extrema. We turn this situation to our advantage by noting that the use of a hill climbing algorithm simultaneously provides the possibility of local bias (via the initial state of the search) while producing a model of the overall behavior of  $f$ .

### 2.1. Partial Derivatives

The first step in developing a hill climbing heuristic is to halve the dimension of the search space. This is made possible by the observation that if the  $x$  were determined, then the appropriate choice of  $y$  is given by the solution of a diagonally dominant tridiagonal linear system. This follows from equating the partial derivatives with respect to  $y_j$  to zero. We have

$$\begin{aligned} \frac{\partial}{\partial y_j} \int_a^b (f(x) - l_K(x))^2 dx &= \int_{x_{j-1}}^{x_j} \frac{\partial}{\partial y_j} \left( f(x) - y_{j-1} - \frac{x - x_{j-1}}{x_j - x_{j-1}} (y_j - y_{j-1}) \right)^2 dx \\ &\quad + \int_{x_j}^{x_{j+1}} \frac{\partial}{\partial y_j} \left( f(x) - y_j - \frac{x - x_j}{x_{j+1} - x_j} (y_{j+1} - y_j) \right)^2 dx \\ &= y_{j-1} \frac{x_j - x_{j-1}}{3} + y_j \frac{2(x_{j+1} - x_{j-1})}{3} + y_{j+1} \frac{x_{j+1} - x_j}{3} \\ &\quad - 2(u_j + v_j), \end{aligned}$$

where

$$u_j = \frac{1}{x_j - x_{j-1}} \int_{x_{j-1}}^{x_j} (x - x_{j-1}) f(x) dx, \quad v_j = \frac{1}{x_{j+1} - x_j} \int_{x_j}^{x_{j+1}} (x_{j+1} - x) f(x) dx.$$

Setting these partials to zero and incorporating the constraints on  $y_0$  and  $y_n$  give rise to the matrix equation

$$Ay = b.$$

Expressing  $A$  in the form

$$\begin{pmatrix} \beta_0 & \gamma_0 & 0 & & & \\ \alpha_1 & \beta_1 & \gamma_1 & * & * & * \\ & & & \alpha_{n-1} & \beta_{n-1} & \gamma_{n-1} \\ & & & 0 & \alpha_n & \beta_n \end{pmatrix}$$

we have

$$\begin{aligned}
\alpha_j &= \begin{cases} x_j - x_{j-1} & \text{if } j = 1, \dots, n-1; \\ 0 & \text{if } j = n, \end{cases} \\
\beta_j &= \begin{cases} 2(x_{j+1} - x_{j-1}) & \text{if } j = 1, \dots, n-1; \\ 1 & \text{if } j = 0 \text{ or } j = n, \end{cases} \\
\gamma_j &= \begin{cases} x_{j+1} - x_j & \text{if } j = 1, \dots, n-1; \\ 0 & \text{if } j = 0, \end{cases} \\
b_j &= \begin{cases} 6(u_j + v_j) & \text{if } j = 1, \dots, n-1; \\ f(a) & \text{if } j = 0; \\ f(b) & \text{if } j = n. \end{cases}
\end{aligned}$$

This linear system allows us to regard the minimization problem as a function of the  $x$  alone. However, when considering partial derivatives with respect to  $x_\ell$ , we must keep in mind that the  $y$  now depend upon the  $x$ . Implicitly differentiating the matrix equation with respect to  $x_\ell$  yields

$$Ay' = b' - A'y,$$

where

$$\begin{aligned}
\alpha'_j &= \begin{cases} 1 & \text{if } j = \ell; \\ -1 & \text{if } j = \ell + 1; \\ 0 & \text{otherwise,} \end{cases} \\
\beta'_j &= \begin{cases} 2 & \text{if } j = \ell - 1; \\ -2 & \text{if } j = \ell + 1; \\ 0 & \text{otherwise,} \end{cases} \\
\gamma'_j &= \begin{cases} 1 & \text{if } j = \ell - 1; \\ -1 & \text{if } j = \ell; \\ 0 & \text{otherwise,} \end{cases}
\end{aligned}$$

and

$$b'_j = \begin{cases} 6 \left( \frac{u_{j+1}}{x_{j+1} - x_j} \right) & \text{if } j = \ell - 1; \\ 6 \left( \frac{v_j}{x_{j+1} - x_j} - \frac{u_j}{x_j - x_{j-1}} \right) & \text{if } j = \ell; \\ -6 \left( \frac{v_{j-1}}{x_j - x_{j-1}} \right) & \text{if } j = \ell + 1; \\ 0 & \text{otherwise.} \end{cases}$$

Having determined the dependence of the  $y$  and  $y'$  on the  $x$ , we can compute the partial derivative of our objective function with respect to  $x_\ell$ :

$$\begin{aligned}
\frac{\partial}{\partial x_\ell} \int_a^b (f(x) - l_K(x))^2 dx &= \frac{\partial}{\partial x_\ell} \int_{x_{\ell-1}}^{x_\ell} \left( f(x) - y_{\ell-1} - \frac{x - x_{\ell-1}}{x_\ell - x_{\ell-1}} (y_\ell - y_{\ell-1}) \right)^2 dx \\
&+ \frac{\partial}{\partial x_\ell} \int_{x_\ell}^{x_{\ell+1}} \left( f(x) - y_\ell - \frac{x - x_\ell}{x_{\ell+1} - x_\ell} (y_{\ell+1} - y_\ell) \right)^2 dx \\
&+ \sum_{j \neq \ell, \ell+1} \int_{x_{j-1}}^{x_j} \frac{\partial}{\partial x_\ell} \left( f(x) - y_{j-1} - \frac{x - x_{j-1}}{x_j - x_{j-1}} (y_j - y_{j-1}) \right)^2 dx \\
&= (y_{\ell+1} - y_\ell) \left\{ 2 \frac{v_j}{x_{j+1} - x_j} - (y_{j+1} - 2y_j)/3 \right\} \\
&+ (y_j - y_{j-1}) \left\{ 2 \frac{u_j}{x_j - x_{j-1}} - (2y_j + y_{j-1})/3 \right\} \\
&+ (x_{j+1} - x_j)(y_{j+1}(2y'_{j+1} + y'_j) + y_j(2y'_j + y'_{j+1}))/3 \\
&+ (x_j - x_{j-1})(y_j(2y'_j + y'_{j-1}) + y_{j-1}(2y'_{j-1} + y'_j))/3 \\
&- 2(y'_j v_j + y'_{j+1} u_{j+1}) - 2(y'_{j-1} v_{j-1} + y'_j u_j) \\
&- 2 \sum_{j \neq \ell, \ell+1} (y'_j - y'_{j-1})(u_j - (x_j - x_{j-1})(y_{j-1} + 2y_j)/6) \\
&- 2 \sum_{j \neq \ell, \ell+1} y'_{j-1}(u_j - (x_j - x_{j-1})(y_{j-1} + y_j)/2)
\end{aligned}$$

## 2.2. Algorithm

In §2.1 we described how to calculate the gradient. Therefore, a standard non-linear optimization routine can be used to determine the  $x$ . In the applications we consider,  $f$  is a piecewise linear function and our implementation is serial. We use a conjugate gradient method (Fletcher-Reeves-Polak-Ribiere; Polak 1971), with a parabolic interpolation method (Brent 1973) for the required line minimizations.

A major source of overhead is the calculation of this gradient. The work involved grows quadratically with the dimension  $n$ ; each component takes  $O(n)$  time. This serves to exacerbate the difficulty of minimizing a function of increasing dimensionality. Asymptotically, this method is impractical, but for small  $n$  the approach works well, providing reasonable global approximations to functions with complicated behavior. Moreover, since the calculation of each component of the gradient need not be tightly coupled with the calculation of other components, an efficient parallel implementation can extend the range of  $n$ .

The expense of gradient and function evaluations may be lessened by pre-calculating the  $L_2$  norm of  $f$ , and the integral and first moment of  $f$  over the subintervals where  $f$  is linear. For

example, let  $f$  be linear over the intervals  $[a_{j-1}, a_j]$  (for  $1 \leq j \leq k$ ) where  $a = a_0 < \dots < a_k = b$ , and define

$$\sigma_0(j) = \sum_{i=1}^j \int_{a_{i-1}}^{a_i} f(x) dx, \quad \sigma_1(j) = \sum_{i=1}^j \int_{a_{i-1}}^{a_i} x f(x) dx.$$

Then the integrals

$$w_j = \int_{x_{j-1}}^{x_j} f(x) dx, \quad m_j = \int_{x_{j-1}}^{x_j} x f(x) dx$$

may be calculated via

$$w_j = \sigma_0(j_{max}) - \sigma_0(j_{min}) + \frac{(a_{j_{min}} - x_{j-1})(f(a_{j_{min}}) + y_{j-1}) + (x_j - a_{j_{max}})(f(a_{j_{max}}) + y_j)}{2}$$

and

$$m_j = \sigma_1(j_{max}) - \sigma_1(j_{min}) + \frac{(a_{j_{min}} - x_{j-1})(f(a_{j_{min}})^2 + f(a_{j_{min}})y_{j-1} + y_{j-1}^2)}{6} + \frac{(x_j - a_{j_{max}})(f(a_{j_{max}})^2 + f(a_{j_{max}})y_j + y_j^2)}{6},$$

where  $j_{min}$  is the minimal  $i$  such that  $x_{j-1} < a_i$ , and  $j_{max}$  is the maximal  $i$  such that  $a_i < x_j$ . The  $u_j$  and  $v_j$  may be calculated via

$$u_j = \frac{m_j - x_{j-1}w_j}{x_j - x_{j-1}}, \quad v_j = \frac{x_{j+1}w_{j+1} - m_{j+1}}{x_{j+1} - x_j}$$

and

$$\int_a^b (f(x) - l_K(x))^2 dx - \|f\|_2 = -2 \sum_{j=1}^n \left\{ y_{j-1}w_j + (y_j - y_{j-1})u_j - \frac{(x_j - x_{j-1})(y_j^2 + y_j y_{j-1} + y_{j-1}^2)}{6} \right\}.$$

We refer to the method of approximation described in this section as *unconstrained placement*. Simple approximation methods do not typically allow the  $x$  to adapt to the peculiarities of the data; the position of the  $x$  must be fixed at the outset. Algorithms which do attempt to place the  $x$  include Friedman's (1988) MARS program and the dynamic programming approach of Bellman and Roth (1969). The problem with methods like MARS is that once an  $x$ -coordinate is chosen, it is not allowed to change; i.e., the influence of  $x_j$  for  $j > k$  on the choice of  $x_k$  is ignored. Consequently, the resulting models can be poor for our application. Although this is not a problem with methods like the dynamic programming approach, they

are even more computationally expensive than ours.<sup>1</sup> Moreover, they do not provide a means of biasing the resulting model (as does our approach via a choice of the initial search state) except by restricting the  $x$  to some predetermined set.

### 3. Constrained Placement

The general problem considered in this part is local approximation. The intent is that the overall behavior of the approximation be controllable while at the same time the resulting model is conservative. In particular, we want to model a function  $f$  by a continuous piecewise linear function  $l$  that has the following properties:

- The model  $l$  interpolates  $f$  at some set of specified locations.
- Let  $\{x_j\}$  denote the endpoints (or *nodes*) of the intervals over which  $l$  is linear. Then  $l$  interpolates  $f$  at each  $x_j$ .
- The model  $l$  is a good approximation to  $f$  in each interval of the form  $[x_{j-1}, x_j]$  in the sense that area is preserved.

The following paragraph formalizes these conditions for the case when  $f$  is itself a piecewise linear function.

Let  $P = \{p_0, \dots, p_r\}$  be a set of points in the domain of a continuous piecewise linear function  $f$  defined over a compact interval  $[a, b]$ . Let  $\{z_i\}$  denote the endpoints (or *nodes*) of the intervals over which  $f$  is linear. A continuous, piecewise linear function  $l$  is an *admissible approximation* to  $f$  if and only if:

- a)  $l(x) = f(x)$  for all  $x \in P$ .
- b)  $l(x) = f(x)$  for all  $x \in N$ , where  $N$  is the set of  $x$ -coordinates of nodes of  $l$ .
- c)  $l$  is the best  $L_2$  approximation to  $f$  over the intervals given by successive elements of  $N$ .

These conditions imply that admissible approximations are contractions; the range of  $l$  is a subset of the range of  $f$ . Moreover, they preserve area; if  $x_j, x_k \in N$ , then

$$\int_{x_j}^{x_k} l(x) dx = \int_{x_j}^{x_k} f(x) dx.$$

Admissible approximations exist, since  $f$  admissibly approximates itself. However, these approximations are not unique, even under the further restriction that the cardinality of  $N$  should be minimal. Although this additional restriction (that  $|N|$  be minimal) defines the approximation problem which we would like to solve, it is not tractable. The method developed in the next section constructs an admissible approximation while attempting to keep the size of  $N$  small, but it does not guarantee the minimality of  $|N|$ .

---

<sup>1</sup>Efficient versions of the dynamic programming approach, such as Hawkins (1972), do not yield continuous models.

### 3.1. Construction

We first consider how conditions b) and c) in the definition of an admissible approximation may be used to locate successive nodes of  $\ell$ . Let the graph of  $f$  over the interval  $[z_0, z_n]$  be represented by the solid line in Fig. 1. Our initial goal is to find  $x_s \in [z_0, z_1]$  and  $x_f \in [z_{n-1}, z_n]$

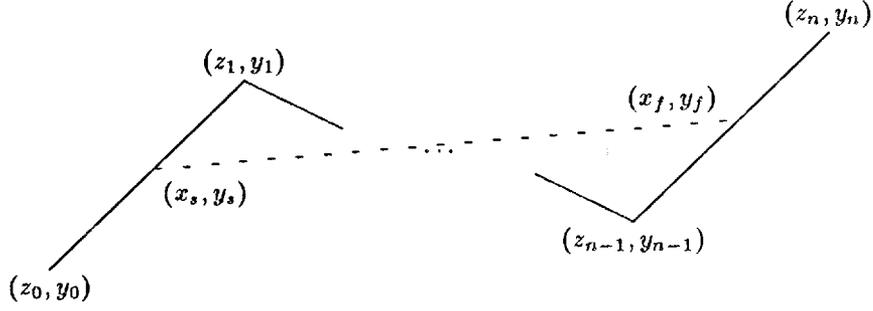


Figure 1: Graphical example of heuristic used to approximate  $\ell$ .

such that the line  $\xi$  that interpolates  $f$  at these two locations is also the best  $L_2$  approximation to  $f$  over  $[x_s, x_f]$ . For now, fix  $x_s \in [z_0, z_1]$ .

Let the point  $(x_f, y_f)$  on the line segment  $\chi$  from  $(z_{n-1}, y_{n-1})$  to  $(z_n, y_n)$  be such that the area under  $f$  from  $x_s$  to  $x_f$  coincides with the area under the line segment  $\xi$  from  $(x_s, y_s)$  to  $(x_f, y_f)$  (the dashed line in Fig. 1). If  $\xi$  exists, it is determined by the relation

$$\begin{aligned} \int_{x_s}^{z_{n-1}} f(x) dx + \frac{x_f - z_{n-1}}{2} \left( 2y_{n-1} + \frac{x_f - z_{n-1}}{z_n - z_{n-1}} (y_n - y_{n-1}) \right) \\ = \frac{x_f - x_s}{2} \left( y_s + y_{n-1} + \frac{x_f - z_{n-1}}{z_n - z_{n-1}} (y_n - y_{n-1}) \right), \end{aligned}$$

which yields

$$\begin{aligned} x_f &= z_{n-1} + \frac{2 \int_{x_s}^{z_{n-1}} f(x) dx - (z_{n-1} - x_s)(y_{n-1} + y_s)}{(z_{n-1} - x_s)(y_n - y_{n-1}) / (z_n - z_{n-1}) - y_{n-1} + y_s} \\ y_f &= y_{n-1} + \frac{x_f - z_{n-1}}{z_n - z_{n-1}} (y_n - y_{n-1}). \end{aligned}$$

These expressions signal the non-existence of  $\xi$  by returning a value for  $x_f$  outside of  $[z_{n-1}, z_n]$ . Moreover, if the denominator (of the expression giving  $x_f$ ) is zero, then  $(x_f, y_f)$  may be taken as any point on  $\chi$  provided that the numerator is also zero. In this case we take  $(x_f, y_f)$  equal to  $(z_n, y_n)$ .

The importance of  $\xi$  is the sense in which it approximates  $\ell$ , the best linear  $L_2$  approximation to  $f$  over  $[x_s, x_f]$ . If  $\alpha$  and  $\beta$  are the coefficients of  $\ell$ , then

$$\frac{\partial}{\partial \beta} \int_{x_s}^{x_f} (f(x) - (\alpha x + \beta))^2 dx = -2 \int_{x_s}^{x_f} (f(x) - \ell(x)) dx = 0$$

and

$$\alpha = \frac{12}{(x_f - x_s)^3} \left( \int_{x_s}^{x_f} x f(x) dx - \frac{x_s + x_f}{2} \int_{x_s}^{x_f} f(x) dx \right).$$

In particular, the area under  $f$  over  $[x_s, x_f]$  also coincides with the area under  $\ell$ . It follows that if the slope of  $\xi$  is  $\alpha$ , then  $\xi$  and  $\ell$  are the same line segment (since then their slopes and areas match). Hence  $\xi$  either approximates  $\ell$  from above in the sense that  $\xi$  has slope greater than or equal to  $\alpha$ , or from below in the sense that  $\xi$  has slope less than or equal to  $\alpha$ .

The utility of this approximation follows from the monotonic dependence of  $x_f$  on  $x_s$ . Since  $(x_s, y_s)$  lies on the line segment from  $(z_0, y_0)$  to  $(z_1, y_1)$ , the quotient rule gives the numerator of the derivative of  $x_f$  with respect to  $x_s$  as

$$y_{n-1} - y_1 + \frac{(y_1 - y_0)(z_1 - z_{n-1} - 1)}{z_1 - z_0} + \frac{y_n - y_{n-1}}{z_n - z_{n-1}}$$

Because this expression is independent of  $x_s$ , it follows that  $x_f$  is a monotonic function of  $x_s$  on  $[z_0, z_1]$ . This implies that if  $x_s$  and  $x'_s$  in  $[z_0, z_1]$  have corresponding points  $(x_f, y_f)$  and  $(x'_f, y'_f)$  on  $\chi$ , then every  $x_s^*$  in the interval determined by  $x_s$  and  $x'_s$  has its corresponding point  $(x_f^*, y_f^*)$  on the line segment from  $(x_f, y_f)$  to  $(x'_f, y'_f)$ . This fact justifies the following bisection algorithm to locate a point  $x_s$  (if it exists) for which  $\xi$  and  $\ell$  are identical:

- I) Determine points  $x_s$  and  $x'_s$  in  $[z_0, z_1]$  such that the corresponding  $\xi$  approximates  $\ell$  from below and  $\xi'$  approximates  $\ell'$  from above.
- II) Let  $x_s^* = (x_s + x'_s)/2$ .
- III) If the corresponding  $\xi^*$  approximates  $\ell^*$  from below, redefine  $x_s$  to be  $x_s^*$ , otherwise redefine  $x'_s$  to be  $x_s^*$ .
- IV) If  $x_s$  and  $x'_s$  are sufficiently close then stop, otherwise go to step II).

Although in presenting the analysis we have considered the existence of  $\xi$  by regarding it as being determined by its left endpoint  $(x_s, y_s)$ , we could have instead proceeded from its right endpoint  $(x_f, y_f)$ . This observation, together with the fact that one endpoint of  $\xi$  varies monotonically with the other, leads to the following heuristic for determining step I) above:

- Determine  $\xi_0$  (if it exists) corresponding to its left endpoint  $(x_s, y_s) = (z_0, y_0)$ .
- Determine  $\xi_2$  (if it exists) corresponding to its left endpoint  $(x_s, y_s) = (z_1, y_1)$ .
- Determine  $\xi_{n-1}$  (if it exists) corresponding to its right endpoint  $(x_f, y_f) = (z_{n-1}, y_{n-1})$ .
- Determine  $\xi_{n+1}$  (if it exists) corresponding to its right endpoint  $(x_f, y_f) = (z_n, y_n)$ .

If there does not exist a pair of approximations from  $\{\xi_0, \xi_2, \xi_{n-1}, \xi_{n+1}\}$  one member of which approximates from above, and the other from below, then instead of searching further, we assume that step 1 is impossible. It turns out that *the property of  $\xi$  being a lower (or upper) approximation of  $\ell$  does not depend monotonically on an endpoint of  $\xi$* , and therefore this heuristic is imperfect.

### 3.2. Algorithm

In order to satisfy condition 1) in the definition of an admissible approximation, we consider the functions  $f_j$  defined by restricting  $f$  to the interval  $[p_{k-1}, p_k]$  (for  $k = 1, \dots, r$ ). Note that if  $\ell_k$  is an admissible approximation to  $f_k$ , then the function  $\ell$  defined locally by  $\ell_k$  will admissibly approximate  $f$ . We may therefore assume without loss of generality that  $P = \emptyset$ .

A simple greedy algorithm may now be used to produce admissible approximations. In what follows, we assume that the  $(z_i, y_i)$  are defined by Fig. 1.

1. Let  $A = a$  and  $B = b$ .
2. Let  $z_0 = A$  and  $z_n = B$  (note that this implicitly defines  $n$  and  $(z_i, y_i)$  for  $i \neq 1, n$ ).
3. Determine  $\xi$  (if it exists) as outlined in steps I) - IV) above.
4. If  $\xi$  was not found, then replace  $B$  by  $z_{n-1}$  and goto step 2.
5.  $\ell$  is defined locally by  $f$  over  $[z_0, x_s]$  and by  $\xi$  over  $[x_s, x_f]$ .
6. If  $x_f = b$  then stop, otherwise let  $A = x_f$ ,  $B = b$ , and goto step 2.

The avidity of this greedy algorithm may be limited by restricting the domain of search from  $[A, B]$  (as initialized by step 2 above) to  $[A, Q]$ . Letting  $Q = A + (b - a)/\lambda$ , where  $\lambda$  is a user-specified parameter, provides some control over the detail of approximation.

We refer to the method of approximation described in this section as *constrained placement*. There does not seem to be any discussion in the literature of the approximations which we have termed admissible.

## 4. Applications

The algorithms described in the previous sections are intended for different purposes and are designed to work together in determining a model for a continuous piecewise linear function  $f$ . The suitability of constrained placement for approximating utilization curves follows from the following design decisions:

- Constrained placement is a conservative method in that it *interpolates* the behavior of  $f$  (every node of the resulting model is a point on the graph of  $f$ ). Interpolation is important, since a utilization model should never indicate that a negative number of processors are in use, nor should it indicate that more processors are active than the system provides.

- Because the intended application is to approximate process utilization curves, and since area under a utilization curve corresponds to work, a natural and useful requirement of any model is that it *preserve area* over each interval of interpolation.
- Our requirement that constrained placement produce a *continuous* model is for simplicity. We believe that continuous models are easier to look at and interpret.
- Finally, the ability of constrained placement to preserve a set  $P$  of user-defined points allows the model to *reflect important features* of the original data. Moreover, automatic identification of underlying trends in processor utilization graphs may rely on using this set  $P$  as a collection of parameters with which to control the model.

In contrast, unconstrained placement does not interpolate, preserve area locally, or maintain a set of user defined points.<sup>2</sup> It is a global approximation technique whose main purpose is to model overall behavior and control complexity. While constrained placement attempts to minimize the number of nodes, it is a local approximation technique which makes no guarantee that the resulting model will be simple. Unconstrained placement, however, is designed with the complexity of the model as a controllable parameter. This makes it particularly useful for investigating a complicated graph with a simple model to bring out global behavior.

We next demonstrate how these two methods may be used interactively in developing a model which is both theoretically sound (in the sense that area is preserved) and reflects the judgement of the user concerning what the important features of  $f$  are. We believe our approximation techniques are particularly well suited for this task.

The graph in Fig. 2 illustrates the ability of unconstrained placement to model global behavior. We took an equal spacing of the  $x$  in the interval  $[a, b]$  as our initial configuration. The dotted function is the input  $f$  which represents the utilization curve of a large time-step algorithm for solving a hyperbolic partial differential equation. The solid function is the resulting approximation  $l_K$  for  $n = 25$ . While this global approximation may look pleasing,

- it does not interpolate,
- it indicates both the utilization of a negative number of processors and the utilization of more processors than are in the system,
- and it does not preserve area over intervals corresponding to successive nodes.

For these reasons, constrained placement will be used to obtain the final model. However, this approximation is useful in that it reveals the basic structure of initialization, three time steps (main computational phase), followed by a degenerate phase and data collection. This approximation also suggests that if high processor utilization is interesting, then there are four clear regions where this takes place. Letting  $P$  be a set of points from the graph of  $f$  reflecting the beginning and ending of these regions, and setting  $\lambda = 1$  to encourage the elimination of other detail, constrained placement results in the following approximation displayed in Fig. 3 This graph illustrates a typical feature of constrained placement; spikes are formed by its greedy

---

<sup>2</sup>A user can *suggest* a set of points by making it the initial configuration for the hill climbing algorithm.

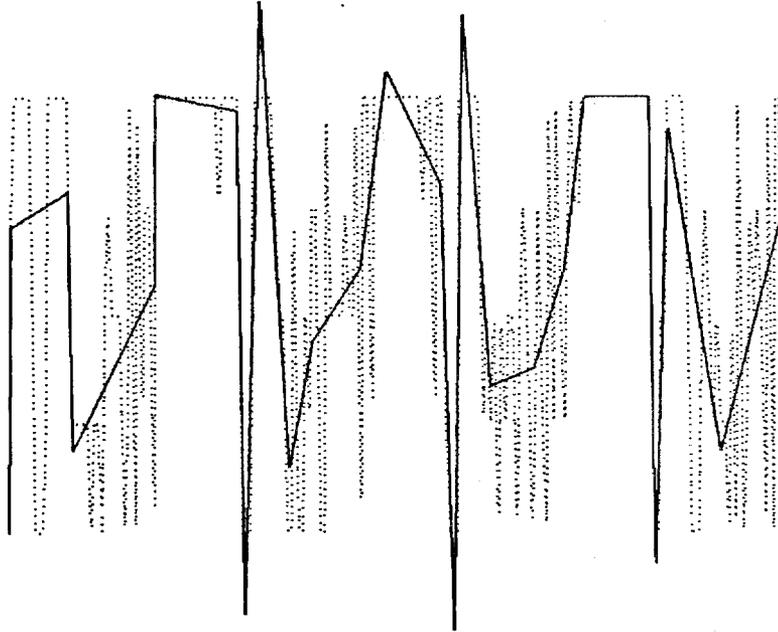


Figure 2: Example fit to processor utilization data using unconstrained placement algorithm.

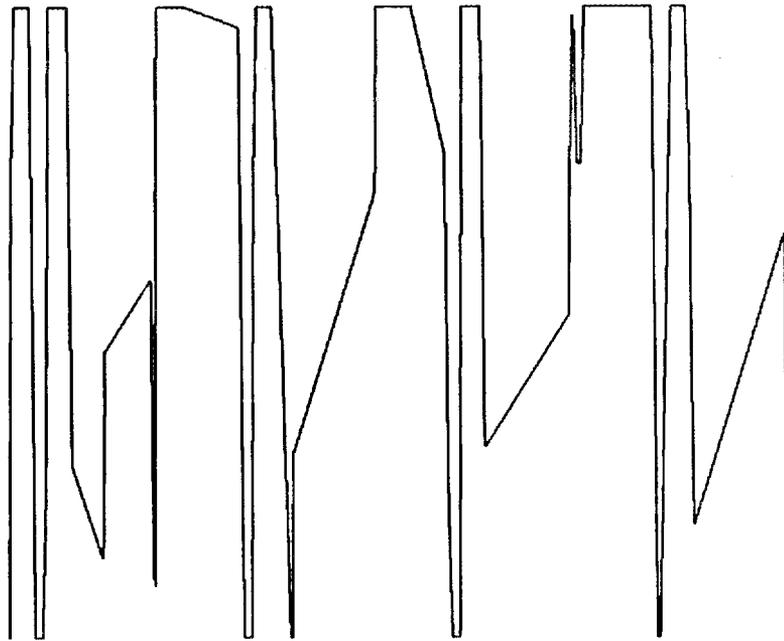


Figure 3: Example fit to processor utilization data using constrained placement algorithm.

algorithm. It is tempting to use unconstrained placement over the intervals surrounding these spikes to smooth them away, but doing so would sacrifice both interpolation and preservation of area. Instead, we use the following idea which requires us only to give up interpolation.

Let  $f$  be an integrable function defined over the real interval  $[a, b]$  and let  $w \in [a, b]$ . Consider the continuous piecewise linear function  $\ell$  determined by the set of nodes

$$\{(a, f(a)), (w, y), (b, f(b))\}.$$

If  $\ell$  preserves area, then we have

$$\begin{aligned} \frac{1}{2}(y + f(a))(w - a) &= \int_a^w f(x) dx \\ \frac{1}{2}(y + f(b))(b - w) &= \int_w^b f(x) dx. \end{aligned}$$

Eliminating  $y$  and rearranging gives

$$2(w - a) \int_w^b f(x) dx + 2(w - b) \int_a^w f(x) dx = (w - a)(b - w)(f(b) - f(a)).$$

This equation can be solved to determine  $\ell$ . Choosing the intervals  $[a, b]$  to surround the spikes in the previous graph and using this method to smooth them away yields the approximation displayed in Fig. 4. This approximation represents our model of processor utilization. We would like to point out that the structure indicated by this model was later verified by examination of the code.

We believe to have developed tools particularly appropriate for modeling processor utilization. The automation of their application is our current focus of research.

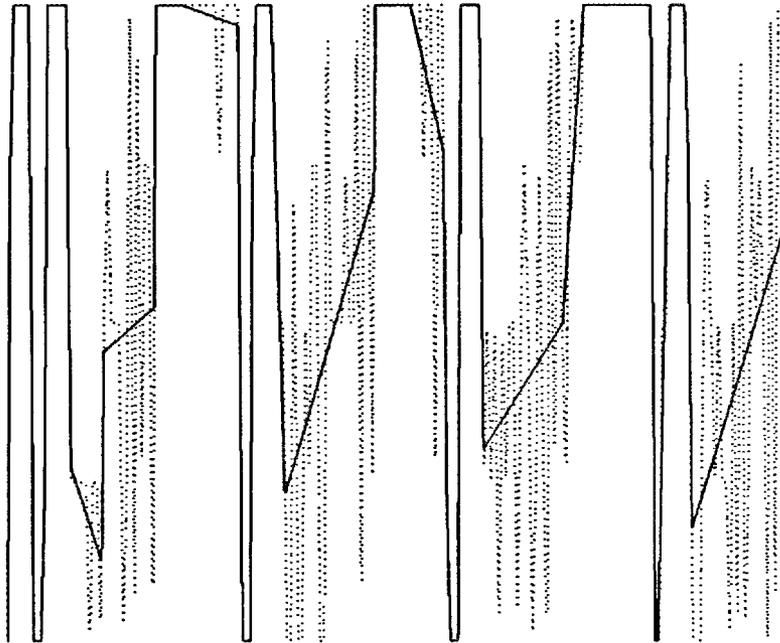


Figure 4: Example fit to processor utilization data using modified constrained placement algorithm.

## References

- [1] R. BELLMAN AND R. ROTH, *Curve fitting by segmented straight lines*, Journal of the American Statistical Association, 64 (1969), pp. 1079–1094.
- [2] R. P. BRENT, *Algorithms for Minimization Without Derivatives*, Prentice - Hall, Englewood Cliffs, NJ, 1973.
- [3] J. H. FRIEDMAN, *Multivariate adaptive regression splines*, Tech. Report No. 102, Laboratory for Computational Statistics, Stanford Univ., CA, 1988.
- [4] D. M. HAWKINS, *On the choice of segments in piecewise approximation*, J. of the Institute of Mathematics and its Applications, 9 (1972), pp. 250–256.
- [5] E. POLAK, *“Computational Methods in Optimization”*, Academic Press, New York, NY, 1971.



**INTERNAL DISTRIBUTION**

- |                          |   |
|--------------------------|---|
| 1. B. R. Appleton        | 22-26. S. A. Raby   |
| 2. E. F. D'Ázevedo       | 27. C. H. Romine  |
| 3. J. J. Dongarra        | 28-32. R. C. Ward   |
| 4. J. B. Drake           | 33-37. P. H. Worley                                       |
| 5. R. E. Flanery         | 38. J. J. Dorning (EPMD Advisory Committee)               |
| 6. G. A. Geist           | 39. R. M. Haralick (EPMD Advisory Committee)              |
| 7-8. R. F. Harbison      | 40. J. E. Leiss (EPMD Advisory Committee)                 |
| 9. M. T. Heath           | 41. M. F. Wheeler (EPMD Advisory Committee)               |
| 10. E. R. Jessup         | 42. N. Moray (EPMD Advisory Committee)                    |
| 11. M. R. Leuze          | 43. Central Research Library                              |
| 12-16. F. C. Maienschein | 44. ORNL Patent Office                                    |
| 17. E. G. Ng             | 45. K-25 Plant Library                                    |
| 18. C. E. Oliver         | 46. Y-12 Technical Library<br>/Document Reference Station |
| 19. G. Ostrouchov        | 47. Laboratory Records - RC                               |
| 20. B. W. Peyton         | 48-49. Laboratory Records Department                      |
| 21. W. M. Post           |   |

**EXTERNAL DISTRIBUTION**

50. Dr. Loyce M. Adams, Department of Applied Mathematics, University of Washington, Seattle, WA 98195
51. Dr. Robert G. Babb, Department of Computer Science and Engineering, Oregon Graduate Center, 19600 N.W. Walker Road, Beaverton, OR 97006
52. Dr. Edward H. Barsis, Computer Science and Mathematics, P. O. Box 5800, Sandia National Laboratory, Albuquerque, NM 87185
53. Dr. David H. Bailey, NASA Ames, Mail Stop 258-5, NASA Ames Research Center, Moffet Field, CA 94035
54. Dr. Robert E. Benner, Parallel Processing Division, 1413, Sandia National Laboratories, Albuquerque, NM 87185
55. Prof. Ake Bjorck, Department of Mathematics, Linkoping University, Linkoping 58183, Sweden
56. Dr. James C. Browne, Department of Computer Sciences, University of Texas, Austin, TX 78712
57. Dr. Bill L. Buzbee, Scientific Computing Division, National Center for Atmospheric Research, P. O. Box 3000, Boulder, CO 80307

58. Dr. Donald A. Calahan, Department of Electrical and Computer Engineering, University of Michigan, Ann Arbor, MI 48109
59. Dr. John Cavallini, Acting Director, Scientific Computing Staff, Applied Mathematical Sciences, Office of Energy Research, U.S. Department of Energy, Washington, DC 20545
60. Dr. Tony Chan, Department of Mathematics, University of California, Los Angeles, 405 Hilgard Avenue, Los Angeles, CA 90024
61. Dr. Jagdish Chandra, Army Research Office, P. O. Box 12211, Research Triangle Park, North Carolina 27709
62. Dr. Melvyn Ciment, National Science Foundation, 1800 G Street, NW, Washington, DC 20550
63. Prof. Tom Coleman, Department of Computer Science, Cornell University, Ithaca, NY 14853
64. Dr. Jane K. Cullum, IBM T. J. Watson Research Center, P. O. Box 218, Yorktown Heights, NY 10598
65. Dr. Paul Concus, Mathematics and Computing, Lawrence Berkeley Laboratory, Berkeley, CA 94720
66. Dr. George Cybenko, Center for Supercomputing Research & Development, 104 South Wright Street, Urbana, IL 61801-2932
67. Ms. Helen Davis, Computer Science Department, Stanford University, Stanford, CA 94305
68. Professor Larry Dowdy, Computer Science Department, Vanderbilt University, Nashville, TN 37235
69. Dr. Iain Duff, CSS Division, Harwell Laboratory, Didcot, Oxon OX11 0RA, England
70. Dr. Stanley Eisenstat, Department of Computer Science, Yale University, P. O. Box 2158 Yale Station, New Haven, CT 06520
71. Prof. David Fisher, Department of Mathematics, Harvey Mudd College, Claremont, CA 91711
72. Professor Geoffrey C. Fox, Physics Department, MS 356-48, California Institute of Technology, Pasadena, CA 91125
73. Dr. Paul O. Frederickson, RIACS, NASA Ames Research Center, Moffet Field, CA 94035
74. Dr. Robert E. Funderlic, Department of Computer Science, North Carolina State University, Raleigh, NC 27650
75. Professor Dennis B. Gannon, Computer Science Department, Indiana University, Bloomington, IN 47401

76. Dr. C. William Gear, Computer Science Department, University of Illinois, Urbana, IL 61801
77. Dr. W. Morven Gentleman, Division of Electrical Engineering, National Research Council, Building M-50, Room 344, Montreal Road, Ottawa, Ontario, Canada K1A 0R8
78. Dr. Alan George, Vice President, Academic and Provost, Needles Hall, University of Waterloo, Waterloo, Ontario, Canada N2L 3G1
79. Dr. Gene Golub, Computer Science Department, Stanford University, Stanford, CA 94305
80. Prof. John L. Gustafson, Ames Laboratory, 236 Wilhelm Hall, Iowa State University, Ames, IA 50011-3020
81. Dr. Eric Grosse, 2C 471, 600 Mountain Avenue, Murray Hill, NJ 07922
82. Dr. Don E. Heller, Physics and Computer Science Department, Shell Development Co., P. O. Box 481, Houston, TX 77001
83. Dr. John L. Hennessy, CIS 208, Stanford University, Stanford, CA 94305
84. Dr. Charles J. Holland, Air Force Office of Scientific Research, Building 410, Bolling Air Force Base, Washington, DC 20332
85. Dr. Robert E. Huddleston, Computation Department, Lawrence Livermore National Laboratory, P. O. Box 808, Livermore, CA 94550
86. Dr. Lennart S. Johnsson, Department of Computer Science, Yale University, P. O. Box 2158 Yale Station, New Haven, CT 06520
87. Dr. Harry Jordan, Department of Electrical and Computer Engineering, University of Colorado, Boulder, CO 80309
88. Dr. Bo Kagstrom, Institute of Information Processing, University of Umea, 5-901 87 Umea, Sweden
89. Professor Malvyn Kalos, Courant Institute for Mathematical Sciences, New York University, 251 Mercer Street, New York, NY 10012
90. Dr. Hans Kaper, Mathematics and Computer Science Division, Argonne National Laboratory, 9700 South Cass Avenue, Argonne, IL 60439
91. Dr. Alan H. Karp, IBM Scientific Center, 1530 Page Mill Road, Palo Alto, CA 94304
92. Dr. Linda Kaufman, Bell Laboratories, 600 Mountain Avenue, Murray Hill, NJ 07974
93. Dr. Robert J. Kee, Applied Mathematics Division 8331, Sandia National Laboratories, Livermore, CA 94550
94. Dr. Kenneth Kennedy, Department of Computer Science, Rice University, P.O. Box 1892, Houston, TX 77001

95. Dr. Tom Kitchens, ER-7, Applied Mathematical Sciences, Scientific Computing Staff, Office of Energy Research, Office G-437 Germantown, Washington, DC 20545
96. Prof. Clyde P. Kruskal, Department of Computer Science, University of Maryland, College Park, MD 20742
97. Prof. Michael Langston, Department of Computer Science, University of Tennessee, Knoxville, TN 37996-1301
98. Dr. Richard Lau, Office of Naval Research, 1030 E. Green Street, Pasadena, CA 91101
99. Dr. Robert L. Launer, Army Research Office, P. O. Box 12211, Research Triangle Park, North Carolina 27709
100. Dr. Scott A. von Laven, Mission Research Corporation, 1720 Randolph Road, SE, Albuquerque, NM 87106-4245
101. Prof. Tom Leighton, Lab for Computer Science, Massachusetts Institute of Technology, 545 Technology Square, Cambridge, MA 02139
102. Dr. Robert Leland, Oxford University Computing Laboratory, 8-11 Keble Road, Oxford, OX1-3QD, ENGLAND
103. Dr. Heather M. Liddell, Director, Center for Parallel Computing, Department of Computer Science and Statistics, Queen Mary College, University of London, Mile End Road, London E1 4NS, England
104. Dr. Joseph Liu, Department of Computer Science, York University, 4700 Keele Street, Downsview, Ontario, Canada M3J 1P3
105. Dr. Franklin Luk, Electrical Engineering Department, Cornell University, Ithaca, NY 14853
106. Dr. Thomas A. Manteuffel, Computing Division, Los Alamos National Laboratory, Los Alamos, NM 87545
107. Dr. James McGraw, Lawrence Livermore National Laboratory, L-306, P. O. Box 808, Livermore, CA 94550
108. Dr. Paul C. Messina, California Institute of Technology, Mail Code 158-79, Pasadena, CA 91125
109. Dr. Cleve B. Moler, MathWorks, 325 Linfield Place, Menlo Park, CA 94025
110. Dr. Dianne P. O'Leary, Computer Science Department, University of Maryland, College Park, MD 20742
111. Professor James M. Ortega, Department of Applied Mathematics, Thornton Hall, University of Virginia, Charlottesville, VA 22901

112. Prof. Merrell Patrick, Department of Computer Science, Duke University, Durham, NC 27706
113. Dr. James C. Patterson, Boeing Computer Services, P.O. Box 24346, MS 7L-21, Seattle, WA 98124-0346
114. Dr. Peter C. Patton, Patton Associates, Inc., 101 International Plaza, 7900 International Drive, Minneapolis, MN 55425
115. Dr. Linda R. Petzold, L-316, Lawrence Livermore National Laboratory, P. O. Box 808, Livermore, CA 94550
116. Dr. Robert J. Plemmons, Departments of Mathematics and Computer Science, North Carolina State University, Raleigh, NC 27650
117. Professor Daniel A. Reed, Computer Science Department, University of Illinois, Urbana, IL 61801
118. Dr. John K. Reid, CSS Division, Building 8.9, AERE Harwell, Didcot, Oxon, England OX11 0RA
119. Dr. John R. Rice, Computer Science Department, Purdue University, West Lafayette, IN 47907
120. Dr. Garry Rodrigue, Numerical Mathematics Group, Lawrence Livermore National Laboratory, Livermore, CA 94550
121. Dr. Donald J. Rose, Department of Computer Science, Duke University, Durham, NC 27706
122. Dr. Ahmed H. Sameh, Center for Supercomputing R&D, 1384 W. Springfield Avenue, University of Illinois, Urbana, IL 61801
123. Dr. Jorge Sanz, IBM Almaden Research Center, Department K53/802, 650 Harry Road, San Jose, CA 95120
124. Dr. Martin H. Schultz, Department of Computer Science, Yale University, P. O. Box 2158 Yale Station, New Haven, CT 06520
125. Prof. Robert B. Schnabel, Department of Computer Science, University of Colorado at Boulder, ECOT 7-7 Engineering Center, Campus Box 430, Boulder, Colorado 80309-0430
126. Dr. David S. Scott, Intel Scientific Computers, 15201 N.W. Greenbrier Parkway, Beaverton, OR 97006
127. The Secretary, Department of Computer Science and Statistics, The University of Rhode Island, Kingston, RI 02881
128. Dr. Horst D. Simon, Mail Stop 258-5, NASA Ames Research Center, Moffett Field, CA 94035

129. Dr. Burton Smith, Teracomputer Company, 400 North 34th Street, Suite 300, Seattle, WA 98103
130. Dr. Marc Snir, IBM T.J. Watson Research Center, Department 420/36-241, P. O. Box 218, Yorktown Heights, NY 10598
131. Prof. Larry Snyder, Department of Computer Science, FR-35, University of Washington, Seattle, WA 98195
132. Dr. Danny C. Sorensen, Department of Mathematical Sciences, Rice University, P. O. Box 1892, Houston, TX 77251
133. Prof. G. W. Stewart, Computer Science Department, University of Maryland, College Park, MD 20742
134. Mr. Steven Suhr, Computer Science Department, Stanford University, Stanford, CA 94305
135. Dr. Wei Pai Tang, Department of Computer Science, University of Waterloo, Waterloo, Ontario, Canada N2l 3G1
136. Dr. Joseph F. Traub, Department of Computer Science, Columbia University, New York, NY 10027
137. Dr. Lloyd N. Trefethen, Department of Mathematics, Massachusetts Institute of Technology, Cambridge, MA 02139
138. Prof. Charles Van Loan, Department of Computer Science, Cornell University, Ithaca, NY 14853
139. Dr. Robert G. Voigt, ICASE, MS 132-C, NASA Langley Research Center, Hampton, VA 23665
140. Dr. Andrew B. White, Los Alamos National Laboratory, P. O. Box 1663, MS-265, Los Alamos, NM 87545
141. Office of Assistant Manager for Energy Research and Development, U.S. Department of Energy, Oak Ridge Operations Office, P. O. Box 2001, Oak Ridge, TN 37831-8600
- 142-151. Office of Scientific & Technical Information, P. O. Box 62, Oak Ridge, TN 37831