ANALYSIS OF FORK-JOIN JOBS USING PROCESSOR-SHARING

C. G. Rommel, D. Towsley, J. A. Stankovic

COINS Technical Report 87-52
May 1987

ANALYSIS OF FORK-JOIN JOBS USING PROCESSOR-SHARING[1]

C. G. Rommel[2], D. Towsley,[3] J. A. Stankovic[3]

University of Massachusetts
Amherst, MA 01003

Abstract

In this paper we derive an expression for the mean response time of a fork-join job in a single server processor-sharing queueing system. We also derive an expression for the mean response time of a fork-join job conditioned on the required service time of the largest task. In our approach a fork-join job is composed of a number of independent tasks which can be scheduled independently of each other. The job is considered complete once the last task completes. Each task service time is assumed to be an exponentially distributed random variable. We provide both lower and upper bounds on mean response time of fork-join jobs. The lower bound to the mean response time of the fork-join problem is very tight when the number of tasks in the job is large (· 7) and/or the server utilization is high. Finally, numerical results are developed that provide various insights such as that fact that processor-sharing scheduling at the job level is better than at the task level.

Key words: Fork-join, job scheduling, lower bound, performance evaluation, processor-sharing, queueing theory, upper bound, and task scheduling.

[1]This work was supported in part by the National Science Foundation under grant MCS-8104203 and by RADC under contract RI-44896X.

[2]Department of Electrical Engineering

[3]Dept. Computer and Information Sciences

1 Introduction

With the advent of programming languages that support parallel programming, (i.e., Concurrent Pascal [Han75], CSP[Hoa85], and Ada [Pyl81]) and multiprocessors, there is increasing interest in modeling the performance of parallel programs. In this paper, we evaluate the performance of a simple parallel program, a *fork-join* on a single server that uses processor-sharing as its scheduling policy. We derive an expression for the mean response time of a fork-join job and both lower and upper bounds on the mean response time. Our lower bound is very tight when the number tasks created by a fork-join program is large. In our problem a fork-join job is composed of a set of tasks each of which can be scheduled independently of the others. The job completes when the last task completes.

There are several reasons why this problem is of interest. First, up to now the literature dealing with the exact performance analysis of parallel programs has focussed on scheduling policies that serve tasks in a first-come first-serve ($FCFS$) manner. Consequently, this paper is a first step toward the analysis of parallel programs under other scheduling policies. We have chosen the processor-sharing discipline because of its usefulness in modeling round robin disciplines that are typically utilized in uniprocessors and multiprocessors.

There is a second practical reason motivating this work. In the future, a large fraction of software will be written using parallel processing constructs such as forks and joins. In most cases, the resulting programs will execute on multiprocessors. However, occasions will occur when the parallel program may be transported to a system containing a single processor. Consequently, it is important to understand the behavior of these parallel programs on a single processor. We will observe from our analysis that a fork-join job incurs a higher mean response time if the basic schedulable entity is a task than if the basic schedulable entity is the job. Consequently, care must be taken in moving parallel programs from multiprocessors to uniprocessors.

As stated before, the literature has not addressed the issue of fork-join jobs in a processor-sharing queueing system. However, processor-sharing and $FCFS$ fork-join job scheduling have been addressed separately.

Bulk arrivals with processor-sharing are examined in [KMR71]. In [KMR71] a general result is given for task response time conditioned on service attained when tasks arrive in bulks. Our approach is an extension of [KMR71] by considering the

average job response time rather than the average task response time.

The behavior of fork-join jobs on both uniprocessors and multiprocessors using a *FCFS* scheduling policy has been evaluated by modeling the system as a bulk arrival queuing system [NTT87]. The behavior of fork-join jobs executing in a system where each task is processed on a dedicated processor has been studied in several papers [FH84], [NT85], [BM85], [TY86]. Some of this work has been extended to parallel programs characterized by arbitrary acyclic precedence graphs [BMT87]. In our approach we extend the uniprocessor work of [NTT87] from the *FCFS* to processor-sharing. In summary, our work may be viewed as an extension of aspects of both [KMR71] and [NTT87] to analyze fork-join jobs under processor-sharing when only one server is used.

The remainder of the paper is structured in the following manner. The model of the system is found in Section 2. The analysis leading to an expression for the mean response time of a job and the conditional response time of a job given the service time of the largest task is contained in Section 3. In Section 3 we also develop both lower and upper bounds for job response time. Section 4 compares various scheduling disciplines. A study of the behavior of the system as a function of different parameters is also given in Section 4. Some concluding remarks are found in Section 5.

2 Model

In our model a job is initially composed of a set of M tasks which arrive to a single server system according to a Poisson arrival process with parameter λ. Here the number of tasks, M, is considered to be a random variable with mean $a = E(M)$. Each task can be executed concurrently with tasks of the same or other jobs. Tasks are assumed to be scheduled independently of each other according to a processor-sharing queueing discipline. The service times of the tasks are assumed to be exponential random variables with mean $1/\mu$. The server utilization is $\rho = \lambda a/\mu$.

It is worth observing that if a job containing M tasks is scheduled as a single entity, its job service time is an M^{th} order Erlang random variable. We shall observe that the variation of processor-sharing where the task is the schedulable entity does not perform as well as the ones where the job is the schedulable entity.

3 Analysis

In this section we determine the average response time of a job conditioned on the service requirement of the longest task and the average response time of a job over all service times. Initially, we shall make no assumptions regarding the task service time distribution. We introduce the following notation:

- X - is the service time of a task with cumulative distribution $B(x)$ $P[X \leq x]$ and mean $1/\mu$,
- $F(x)$ - is the cumulative distribution of the remaining service time where $F(x) = 1 - B(x)$,
- $\hat{X}_i$ - the service time of the i^{th} largest task in a job, i.e., $\hat{X}_1 \geq \hat{X}_2 \geq \cdots \geq \hat{X}_M$,
- Y - is the service time of the largest task in a job, Y $\hat{X}_1$,
- W - is the mean response time of a task in the job,
- t - is the mean response time of the job such that t $E(\max_{1 \leq i \leq M}\{W_i\})$,
- $w(x)$ - is the mean response time of a task in the job conditioned on the service time, X x, i.e., $w(x)$ $E(W|X$ $x)$.
- $t(x)$ - is the mean response time of the job conditioned on the service time of largest task, Y x, i.e., $t(x)$ $E(T|Y$ $x)$,
- $M(x)$ - number of incomplete tasks in a job after each task has been entitled to x units of service, i.e., $M(x)$ m *iff* $\hat{X}_m > x \geq \hat{X}_{m+1}$, $1 \leq m \leq M$ and $\hat{X}_1 > x$,
- $t_m(x)$ $E(T|Y$ x, M $m)$,
- $w_m(x)$ $E(W|Y > x, M$ $m)$,
- $w'(x)$ $dw(x)/dx$,
- $w'_m(x)$ $dw_m(x)/dx$,
- $t'(x)$ $dt(x)/dx$.

We next derive an integro-differential equation for the mean job response time, establish closed form expressions for $t_m(x)$ and t when X is an exponential random variable, and establish bounds for these quantities.

3.1 Derivation of the Processor-Sharing Integro-Differential Equation

Our derivation of the task-sharing integro-differential equation follows the development of [KMR71] and relies on the feedback approach to processor-sharing presented in [KC67]. Our system is composed of a single queue and a processor-sharing server. Jobs enter at the tail of the queue and are split into tasks. Figure 1 gives a diagram of the queueing system when a particular job arrives. In the analysis to follow we will examine the progress of this job through the system. This job is referred to as the *tagged job.*

Each task receives a quanta of service when it reaches the server. If the attained service of the task is less than the required service, the task is placed at the end of the queue. Otherwise, the task exits. In this approach we first analyze the response time of a fork-join job when the scheduler is a round-robin discipline with a time slice, q. Then we consider the limiting case of round-robin as the time slice quantum approaches zero.

First, we will define the following in the context of the round-robin discipline :

- n_i - the mean number of tasks in the system when the largest task of the job receives it's i^{th} quantum of service,
- $N(x)$ - the task density of the system given x units of attained service,
- t_i - is the average delay between the $(i-1)^{th}$ quantum of service and the i^{th} quantum of service for the largest task including the actual quantum of service,
- σ_i - the probability that a task which has received iq units of service will require more than $(i+1)q$ seconds of service, and
- ζ_i - the probability that a task of a job which has received iq seconds of service will require more than $(i+1)q$ seconds of service given that the largest task still requires additional service.

The average delay until the largest task of the job receives its first quantum of service is simply

$$t_1 \quad \sum_{j \quad 0}^{\cdot\cdot} n_j q + (m \quad 1)q + q . \tag{1}$$

We assume here that the largest task enters at the end of the queue. This ordering is not important since we consider the limit as $q \quad \cdot 0$. Because of this limit, any existing tasks share the processor immediately. The first term in the above equation represents the delay due to tasks ahead of the newly arriving job. The second term models the effect of the delay due the members of the newly arriving job in front of the largest task. The last term gives the delay due to the first quantum of service of the largest task.

By induction, the mean time for the largest task to receive its i^{th} quantum of service is given as the sum of four terms.

$$\begin{aligned} t_i \quad & \sum_{j \quad 0}^{\cdot\cdot} n_j \sigma_j \sigma_{j \quad i+1} \ldots \sigma_{j+i \quad 2} q + \\ & \sum_{j \quad 1}^{i \quad 1} \lambda a t_j \sigma_0 \sigma_1 \ldots \sigma_{i \quad j \quad 2} q + \\ & (m \quad 1) q \prod_{j \quad 0}^{i \quad 2} \varsigma_j + q . \end{aligned} \tag{2}$$

The first term is associated with tasks found when the job arrived at the site; the next term is associated with tasks which arrived after the job; the third term is due to the tasks of the job which remain after receiving i quanta of service; and q is the quantum of service that the largest member receives. Following Kleinrock's development we divide by q leaving :

$$\begin{aligned} t_i/q \quad & \sum_{j \quad 0}^{\cdot\cdot} n_j \sigma_j \sigma_{j \quad i+1} \ldots \sigma_{j+i \quad 2} + \\ & \sum_{j \quad 1}^{i \quad 1} \lambda a t_j \sigma_0 \sigma_1 \ldots \sigma_{i \quad j \quad 2} + \\ & (m \quad 1) \prod_{j \quad 0}^{i \quad 2} \varsigma_j + 1. \end{aligned} \tag{3}$$

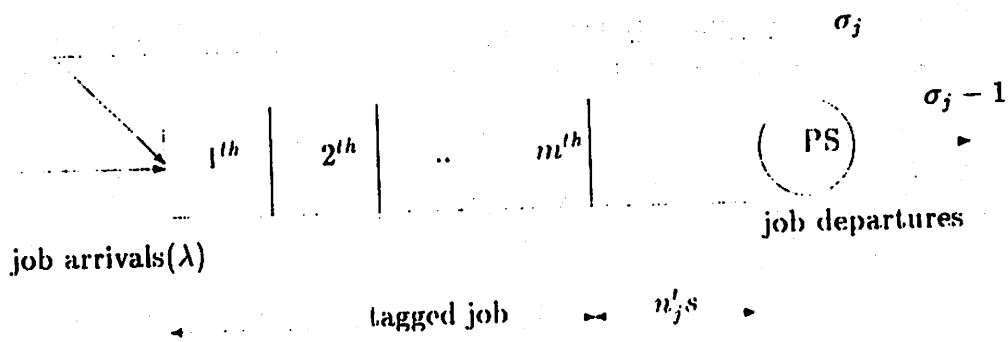

Figure 1: Process-Sharing Diagram after the Arrival of a Tagged Job

In the limit as $i \to \infty$, $j \to \infty$ such that

$iq \to x$, $jq \to y$, and $q \to 0$, we obtain the following limits:

$$t_i/q \to t'_m(x), \tag{4}$$

$$t_j \to t'_m(y)dy, \tag{5}$$

$$n_i \to N(y)dy, \tag{6}$$

$$\sigma_j\sigma_{j+1}\ldots\sigma_{j+i-2} \to (1 - B(y+x))(1 - B(y)), \tag{7}$$

$$\sigma_0\sigma_1\ldots\sigma_{i-2} \to 1 - B(x-y), \tag{8}$$

$$1 + (m-1)\sum_{j=0}^{i-2}\zeta_j \to E(M(x)|Y = x). \tag{9}$$

$$\tag{10}$$

By taking the limits of both sides of equation (3) as $q \to 0$, we obtain

$$t'_m(x) = \int_0^\infty N(y)\frac{(1 - B(y+x))}{(1 - B(y))}dy + \lambda a\int_0^x t'_m(y)(1 - B(x-y))dy + E(M(x)|Y = x). \tag{11}$$

By noting that the average density of tasks [KC67] is

$$N(y) = \lambda a[1 - B(y)]w'(y) \tag{12}$$

we obtain our first main result.

$$t'_m(x) = \lambda a\int_0^\infty w'_m(y)(1 - B(y+x))dy + \lambda a\int_0^x t'_m(y)(1 - B(x-y))dy + E(M(x)|Y = x). \tag{13}$$

We remind the reader that the above equation makes no assumption regarding the nature of the service time distribution.

3.2 Fork-Join Jobs with Exponential Service Times and Processor-Sharing

If we assume exponential service times with mean $1/\mu$ for each task, then several simplifications occur. First, since the cumulative distribution is now exponential, we have

$$(1 \quad B(y \mid x)) \quad e^{\mu(y \mid x)}, \tag{14}$$

$$(1 \quad B(x \quad y)) \quad e^{\mu(x \quad y)}. \tag{15}$$

From the results of [KMR71] we know that

$$w'_m(y) \quad \frac{1}{(1 \quad \rho)} + \frac{(a \quad 1)(2 \quad \rho)e^{\mu(1 \quad \rho)y}}{2(1 \quad \rho)}. \tag{16}$$

Then by integration we obtain

$$\lambda a \int_0^{\cdot\cdot} w'_m(y)(1 \quad B(y \mid x))dy \quad \frac{0.5\rho(a \mid 1)e^{\mu x}}{(1 \quad \rho)}. \tag{17}$$

Using the exponential assumption we now have

$$\lambda a \int_0^x t'_m(y)(1 \quad B(x \quad y))dy \quad \rho\mu \int_0^x t'_m(y)e^{\mu(x \quad y)}dy. \tag{18}$$

Let us now focus on $E(M(x)|Y \quad x)$. By using the definition of the expectation of a random variable we have

$$E(M(x)|Y \quad \cdot x) \quad \sum_{k \quad 1}^{m} kP(M(x) \quad k|Y \quad \cdot x). \tag{19}$$

From the definition of conditional probability we know that

$$P(M(x) \quad k|Y \quad \cdot x) \quad \frac{P(M(x) \quad \cdot k \text{ and } Y \quad \cdot x)}{P(Y \quad \cdot x)}. \tag{20}$$

Since the event $(M(x) \cdot k)$ '_ event $(Y \quad x)$ $k \_ 1$, we have

$$P(M(x) \quad k \text{ and } Y \cdot x) \quad \begin{cases} 0 & ,k \quad 0 \\ P(M(x) \cdot k) & ,0 \cdot k \cdot m. \end{cases} \tag{21}$$

By observing that $M(x)$ is a binomial random variable, we have

$$P(k) \quad \binom{m}{k} |1 \quad F(x)|^{m \ k} |F(x)|^k, k \cdot 1. \tag{22}$$

By direct substitution we have

$$E(M(x)|Y \quad x) \quad \frac{\sum_{k}^{m} k\binom{m}{k}|1 \quad F(x)|^{m \ k}|F(x)|^k}{P(Y \cdot x)}. \tag{23}$$

If we now consider the denominator of the above expression, we see that by the basic probability axiom and substitution we have

$$P(Y \cdot x) \quad 1 \quad P(Y \quad x) \quad 1 \quad (1 \quad e^{\ \mu x})^m. \tag{24}$$

Then noting that the numerator is the expectation of a binomial random variable with $q \cdot |1 \quad F(x)|$ and $p \quad F(x)$ and that $F(x) \quad e^{\ \mu x}$, we have our desired result:

$$E(M(x)|Y \cdot x) \quad \frac{me^{\ \mu x}}{1 \quad (1 \cdot e^{\ \mu x})^m}. \tag{25}$$

When we combine these equations together, we form the integro-differential equation for the job response time for exponential service requirements.

$$\begin{aligned} t'_m(x)) \quad & \frac{0.5\rho(a \ 1)}{(1 \quad \rho)} e^{\ \mu x} \ 1 \\ & e^{\ \mu x}\rho\mu \int_0^x t'_m(y)e^{\mu y}dy \ 1 \\ & \frac{me^{\ \mu x}}{1 \quad (1 \quad e^{\ \mu x})^m}. \end{aligned} \tag{26}$$

Equation (26) is our second important result.

To eliminate the exponential product in the integral of equation (26) we make the following substitution

$$t'_m(x) \quad R'(x)e^{\ \mu x}. \tag{27}$$

By some algebra we may formulate the above equations into the following linear differential equation,

$$R'(x) \quad \rho\mu R(x) \cdot \quad \rho\mu R(0) + \frac{0.5\rho}{(1 \cdot \rho)}(a + 1) + (1 \quad (1 \quad e^{-\mu x})^m)^{m} \cdot \tag{28}$$

Equation (28) may be solved either analytically or numerically by standard techniques. To determine the average response time of the job response time, $t_m(x)$ we integrate $R'(x)e^{-\mu x}$. Therefore,

$$t_m(x) \quad \int_0^x R'(y)e^{\ \mu y}dy\,. \tag{29}$$

We now solve for the average response time of the job by conditioning on Y. The distribution of Y for the largest task is the same as the maximum order statistic of the exponential distribution. It is given as

$$f_{Y|M}(x|m) \quad m\mu(1 \quad e^{\ \mu x})^{m-1}e^{\ \mu x}. \tag{30}$$

Then the average job response time, t is simply

$$t \quad \int_0^{\infty} t_m(y)f_{Y|M}(y|m)dy\,. \tag{31}$$

This completes our general analysis for the response time of a fork-join job using processor-sharing scheduling with exponential service requirements.

3.3 Closed Form Solution to the Fork-Join Job Problem

There are several ways to solve for equation (28). One way is to use numerical methods to determine R, $t'_m(x)$, and $t_m(x)$. We use a numerical appr... [illegible] the section 4 because of its simplicity. However, to define the contributions of various parameters to the response time of the job, we derive a closed form solution to equation (28) in this section.

To determine a closed form expression for the response time, we expand $m/(1 - (1 - e^{-\mu x})^m)$ in terms of a sum of exponentials such that

$$m/(1 \quad (1 \quad e^{-\mu x})^m) \quad e^{\mu x} + \sum_{j=0}^{\infty} c_{mj} e^{-j\mu x}. \tag{32}$$

Appendix A gives a development for equation (32) and describes how the parameters, $c_{mj}, j = 0, 1, ...\infty$ may be obtained. This development is based on a partial fraction expansion approach in order to avoid instability problems. Next, we observe that equation (28) is a simple first order differential equation. The homogeneous solution is

$$R_h(x) \quad Ce^{\rho x}, \tag{33}$$

where C is a integration constant. To find the particular solution we use the expansion of $m/(1 \quad (1 \quad e^{-\mu x})^m)$ given by equation (32).

The total expansion of the right hand side of equation (28) is

$$K + c_{m0} + e^{\mu x} + \sum_{j=1}^{\infty} c_{mi} e^{-j\mu x} \tag{34}$$

where K is the constant term from equation (28) and c_{mi} are the constants of the expansion of $m/(1 \quad (1 \quad e^{-\mu x})^m)$. The particular solution is given by

$$R_p(x) \quad \frac{K + c_{m0}}{\rho\mu} + \frac{e^{\mu x}}{\mu(1 \quad \rho)} \quad \sum_{j=1}^{\infty} \frac{c_{mj} e^{-j\mu x}}{\mu(j + \rho)} \tag{35}$$

and the complete solution is then

$$R(x) \quad Ce^{\rho x} + \frac{K + c_{m0}}{\rho\mu} + \frac{e^{\rho x}}{\mu(1-\rho)} \quad \sum_{j=1}^{\infty} \frac{c_{mj} e^{-j\mu x}}{\mu(j+\rho)}. \tag{36}$$

By using the procedures in the previous section, we obtain the solution for the response time conditioned on the service requirement

$$t_m(x) = \frac{x}{1-\rho} + \frac{\rho C'}{(1-\rho)}(1 - e^{-(1-\rho)\mu x}) + \frac{1}{\mu} \sum_{j=1}^{\infty} \frac{j c_{mj}(1 - e^{-(j+1)\mu x})}{(j+1)(j+\rho)} \tag{37}$$

with the initial condition that $t_m(0) = 0$. The factor $\rho C'$ may be determined from the initial conditions on the equation (26) and the derivative of $t_m(x)$. The result is

$$\mu\rho C' = \frac{1}{1-\rho}(0.5\rho(a+1) - 1) + m \sum_{j=1}^{\infty} \frac{j c_{mj}}{(j+\rho)}. \tag{38}$$

We can also determine the average response time of the job over all service times. This result is given as

$$t = \frac{1}{\mu}\left[H_m + \frac{\rho C'}{(1-\rho)}\left(1 - \frac{m!}{\prod_{j=1}^{m}(1+j-\rho)}\right) + \sum_{j=1}^{\infty} \frac{j c_{mj}}{(j+1)(j+\rho)}\left(1 - \frac{m!}{\prod_{i=1}^{m}(1+j+i)}\right)\right] \tag{39}$$

where H_m is the harmonic series($H_m = \sum_{i=1}^{m} 1/m$). Equation (39) is valid for $\rho > 0$. An expression can be developed for $\rho = 0$ yielding

$$t = \frac{m}{\mu} \tag{40}$$

Before continuing, we investigate the term

$$\frac{1}{(1-\rho)}\left(1 - \frac{m!}{\prod_{j=1}^{m}(1+j-\rho)}\right). \tag{41}$$

in equation (39).

We observe that it can be expressed as

$$\frac{\sum_{j=0}^{m-1} d_j \rho^j}{\prod_{j=1}^{m}(1+j-\rho)} \tag{42}$$

where the d_j terms are the result of synthetic division by $(1-\rho)$ into $(1-\prod_{j=1}^{m}(1+j-\rho))$. This implies that our result depends on $1/(1-\rho)$ and not $1/(1-\rho)^2$.

We also observe that equation (42) converges to H_m as ρ approaches one. With these facts in mind we may make the following observations:

1. the mean response time of a job grows in a with slope $\frac{1}{\mu}(1+\frac{0.5\rho}{(1-\rho)})(1-\prod_{j=1}^{m}(1+j-\rho))$ and
2. the mean response time of a job grows nonlinearly in m as a function of ρ.

3.4 Bounds on Job Mean Response Time

In this section we develop bounds on the average response time of fork-join jobs. These bounds are important since the evaluation of the exact mean response time becomes computationally complex as m grows.

We obtain a lower bound, $R^{lb}(x)$, by bounding the $m/(1-(1-e^{-\mu x})^m)$ term of the right hand side of equation (28) by m. The proof of this may be found in [Rom87]. This bound becomes tight when either $m \to \infty$, or $\mu \to 0$. The resulting differential equation

$$R'^{lb}_m(x) = \frac{0.5\rho(a+1)}{(1-\rho)}e^{-\mu x} + e^{-\mu x}\rho\mu\int_0^x R'_m(y)e^{\mu y}dy + m \tag{43}$$

can be easily solved using the methods of section 3.3 to yield

$$R^{lb}_m(x) = \rho(x+\frac{1-e^{-(1-\rho)\mu x}}{(1-\rho)}). \tag{44}$$

Removal of the conditioning on $Y = x$ yields

$$R^{lb} = \frac{1}{\mu}+\frac{0.5\rho(a+1)}{\mu(1-\rho)^2} + \frac{m}{(1-\rho)}-\left|1-\frac{m!}{\prod_{j=1}^{m}(1+j-\rho)}\right|. \tag{45}$$

Equation (45) gives similar insights into processor-sharing job response time as does equation(39). We see similar dependence of the response time on a, ρ, and m.

An upper bound may be derived in a similar fashion to the lower bound. The bound itself results from bounding the $me^{-\mu x}/(1-(1-e^{-\mu x})^m)$ term of the right hand side of equation (26) by m. This upper bound becomes tight when either the task size, $m \to 1$, or $\mu \to 0$. The results for the conditioned mean response time and unconditional mean response time are given without proof :

$$t_m^{ub}(x) \quad \rho(\quad \frac{1-e^{-(1-\rho)\mu x}}{(1-\rho)} \quad ; \quad \frac{mx}{1-\rho} \tag{46}$$

where

$$\rho\mu(\quad \frac{1}{1-\rho}(0.5\rho(a+1)) \quad m \quad \frac{\rho}{1-\rho}; \tag{47}$$

and

$$t^{ub} - \frac{1}{\mu}\left|\frac{mH_m}{(1-\rho)} + \left(\frac{0.5\rho(a+1)}{(1-\rho)^2} \quad \frac{\rho m}{(1-\rho)^2}\right)\left(1 \quad \frac{m!}{\prod_{j=1}^{m}(1+j-\rho)}\right)\right|. \tag{48}$$

This completes our treatment of the job response time.

4 Numerical Results

In this section we give numerical results for the scheduling algorithm we have just analyzed - the processor-sharing(PS) discipline in which the task is the basic schedulable entity. The average response time of the job for this algorithm is given by equation (31). We refer to this model as the $M/G/1 \quad PS \quad TASK$ model. Our analysis will be based on a numerical solutions of equation(31). In section 4.1 we define the details of the solution technique in solving equation (31). Section 4.2 compares the average response time of two algorithms: task scheduling using processor-sharing to job scheduling using processor-sharing. Section 4.3 provides a comparison of task scheduling with processor-sharing to job scheduling using $FCFS$. Section 4.4 examines the lower bound given in equation (45) and the upper bound given by equation(48). Finally, Section 4.5 describes results for task scheduling under processor-sharing in which average number tasks per job at the server is not equal to the number of tasks in a particular job, i.e. $a \neq m$.

4.1 Numerical Details

To investigate the issues in sections 4.2 through 4.5 inclusive we examine response time given by equation (28) for jobs from 1 to 8 tasks under utilizatic · f -: 0.1 to 0.9 with a average service time of 10000 milliseconds. We obtain the mean response time of the $M/G/1$ PS $TASK$ system using a fourth order Runge-Kutta method for solving equation (31). We use a increment size of 200 milliseconds and 4000 points. When we consider processor-sharing with job scheduling, $FCFS$ with job scheduling, and average task response time under $M/G/1$ PS $TASK$, we also assume that all jobs consist of exactly the same number of tasks, *i.e.* a m. This is done for ease of comparison only. In section 4.5 we consider the case when $a \neq m$ using the same parameters above with a varying independently of m.

4.2 Task Scheduling and Job Scheduling under Processor-Sharing

The first scheduling algorithm, which we compare to the $M/G/1$ PS $TASK$, is the PS discipline where the job is the smallest schedulable entity. The fact that the job might be composed of tasks is immaterial to the scheduler. We refer to this as the $M/G/1$ PS JOB model. The response time of the job is given by

$$t \quad \frac{m}{\mu}\left(\frac{1}{1-\rho}\right). \tag{49}$$

Figure 2 plots the ratio of the average response time of the $M/G/1$ PS $TASK$ system scheduling to the $M/G/1$ PS JOB system for jobs of 1, 3, 5, and 7 tasks. Several observations can be made from our data. First, we see that as the utilization, ρ, approaches zero the performance of the $M/G/1$ PS $TASK$ system approaches the performance of the $M/G/1$ PS JOB system independent of the number of tasks in a job. This is expected since no queueing occurs in both conditions. In addition, the curve representing a job with only 1 task (m - 1) in Figure 2 shows that the ratio of the $M/G/1$ PS JOB system to the $M/G/1$ PS $TASK$ system equals 1 independent of the number of tasks. In general, we conclude that the performance of the two systems approach each other as m · 1. This result is also expected. However, as the number of tasks in a job increases and the utilization increases, then the $M/G/1$ PS JOB system gives better performance than the $M/G/1$ PS $TASK$ system. This leads to

an important observation that *processor-sharing scheduling at the job level is better than that at the task level.* This is can be explained since the amount of service given to a job in the $M/G/1 \quad PS \quad TASK$ system is a linear function of the number of tasks. Consequently, large jobs get preferential treatment. As an example, when $m \quad 7$ and $\rho \quad .9$, we see the job response time is nearly 50% higher for the $M/G/1 \quad PS \quad TASK$ than the $M/G/1 \quad PS \quad JOB$ system. Moreover, we see an increase in the ratio of response times with the utilization, ρ. This is also expected since as ρ increases small tasks must share the processor with a larger number of tasks over a longer period of time. Thus, a job composed of small tasks are delayed longer in the task scheduling approach than the job scheduling approach.

4.3 Task Scheduling under Processor-Sharing and Job Scheduling under $FCFS$

The second numerical study we perform is to compare task scheduling under processor-sharing to job scheduling under $FCFS$. Although the tasks are scheduled sequentially, each task service time is exponential. In this model the service time is Erlangian. We refer to this as the $M/G/1 \quad FCFS$ model. The expected delay is given in [All78] as

$$t \quad \frac{m}{\mu}(1 + \rho^{0.5(1 + 1/m)}). \qquad (50)$$

Figure 3 displays the ratio between the job response time under task scheduling under processor-sharing to the job response time under $FCFS$. We see that even for low utilization, the $M/G/1 \quad FCFS$ system performs better than processor-sharing approach. Only when the utilization approaches zero do both algorithms have the same performance. We see that ratios are greater than for the simple task and job scheduling algorithms. This is expected since the coefficient of variation of the service time of the $M/G/1 \quad FCFS$ system is less than the corresponding $M/G/1 \quad PS \quad JOB$ system. Therefore, it is expected to be less than the $M/G/1 - PS - TASK$ system. To continue the example from section 4.2, we see that at $m = 7$ and $\rho \quad .9$ the job response time is nearly 200% more for the $M/G/1 - PS - TASK$ than the $M/G/1 \quad FCFS$ system.

4.4 Lower and Upper Bounds on Job Response Time

Now we want to compare our bounds to the exact solution. Figure 4a shows the tightness of the lower bound equation (45) developed in the previous section. This figure gives the difference between the lower bound and the exact solutions as a percentage of the latter. We give curves for the task number, m, of 1, 3, 5, and 7. We see that the lower bound approaches the exact solution as $\rho \to 1$. We also observe that the bound as $\rho \to 1$ is independent of the task count. This is expected since the dominant factor is the utilization of the server. In addition, we observe that for tasks counts of 5 or greater our bound is within 15% of the exact solution. For task counts of 7, our bound is nearly within 10% of the exact solution where the utilization is low ($\rho < 0.7$) and within 5% for high utilization ($\rho > 0.7$). It appears that the bound is close enough for $m > 7$ to be used as an approximate solution. Figures 4b, 4c and 4d plot the response time of the job using the lower bound, the exact solution and the upper bound. In 4b we use $m = 2$. In 4c we use $m = 5$. Finally, in 4d we use $m = 7$. We observe that both bounds are good for $m = 2$. However, the upper bound becomes worst as we increase m whereas the lower bound becomes tighter as we increase m. As an example observe the difference between the lower bound and exact solutions for $m = 7$ and $\rho = 0.8$. There is clearly little difference.

4.5 Job Response Time with $a \neq m$

The next experiment considers the effect of the overall average number of tasks, a, on the response time of the tagged job. The reader should recall from section 3.1 that a tagged job is simply a particular job with a given m. The number of tasks in this job may be different from the average number of tasks for all jobs at a server since a need not equal m. In this experiment we consider the number of tasks of the tagged job to take on values of 1, 3, 5, and 7 tasks per job while varying the value of the overall average number of tasks per job to 1 to 7. Figures 5a, 5b, 5c, and 5d give the response times in milliseconds for values of $\rho = 0.1, 0.5, 0.7$ and 0.9 with a as the independent parameter. As observed in Section 3.3, we see that a plays linear role in response time for a give value of ρ. In fact, we can determine the slope of the response time curve from equation(39). It is given as $\frac{1}{\mu}(1 - \frac{0.5\rho}{\rho})^2(1 - \prod_{?}^{m} \frac{m!}{(1+)} \rho)$.

Figures 6a, 6b, 6c and 6d plot the response time as a function of m instead of a. We see a clear nonlinear dependence on m as expected from equation(39). If we observe

Figure 6d, we see the most nonlinear dependence. However, after about $\rho = 0.5$, we observe that the increase in response time is approximately a linear function of m. This is expected since our lower bound becomes both tight and linear as $\rho \to 1$.

5 Conclusions

In summary, we have developed an exact solution for solving the fork-join processor-sharing with a single server queue. We have a solution for the rate of the response time conditioned on service time requirements. We also have a numerical solution for the response time for fork-join jobs with exponential service times. We have developed both lower and upper bounds for the fork-join processor-sharing model.

Our results demonstrate that scheduling fork-join jobs under processor-sharing should be done at the job level and not at the task level. This consideration is clearly important for single server applications. It is also important for multi-server systems applications when these systems degrade to single server systems. In these applications, jobs may be split for scheduling at different servers. However, the tasks of a job may be forced to remain together at a particular site due to possible faults in a communication network, network scheduling problems, or remote server overloading. From our observations, in such circumstances the job should not be split at the site. In fact, if the tasks of the job are grouped together, the overall system performance is improved. In conclusion, fork-join jobs should be carefully scheduled when moved from a multiprocessor system to a single processor system when processor-sharing is used.

Acknowledgments: We would liked to acknowledge Professor Gerald Giessert's assistance is obtaining equation (25).

Appendix A Exponential Series For Equation(25)

We consider a special approach to the expansion of equation(25) to prevent stability problems which arise when using the standard geometric expansion. Our expansion begins by noting that

$$(1 \quad (1 \quad e^{\mu z})^m) \qquad z^m \quad 1 \tag{51}$$

$$where z \quad 1 \quad e^{\mu z}. \tag{52}$$

We also know that

$$(1 \quad (1 \quad e^{\mu z})^m) \qquad m \prod_{j \ 0}^{m} z \quad r_j \tag{53}$$

$$where r_j \quad e^{2\pi j/m} j \quad 0, 1, 2, \ldots, m \quad 1 \ and \tag{54}$$

$$i \quad \sqrt{} \quad 1. \tag{55}$$

By partial fraction expansion and substitution we have

$$(1 \quad (1 \quad e^{\mu z})^m) \qquad e^{\mu z} \ \sum_{j \ 1}^{m} \frac{g_j}{1 \quad r_j \quad e^{\mu z}} \tag{56}$$

$$where g_j \qquad m \prod_{k \ 0, k \neq j}^{m} z \quad \frac{1}{r_k} \Big|_{z \ r_j}. \tag{57}$$

The values for g_j are complex numbers in general. To expand the above equation in terms of exponentials we formulate the term associated with the j^{th} root term as

$$\frac{g_j/(1 + b_j \quad r_j)}{1 \quad (b_j + e^{\mu z})/(1 + b_j \quad r_j)}. \tag{58}$$

The value of b_j must be inserted so that the following series expansion converges as a geometric series. It can be shown that convergence occurs when $cos(2\pi j/m) > 0.5/(b_j + 1)$. We may now find the c_{mj} terms. To do this now formulate algorithm, *Algorithm* 1.

Algorithm 1: Determination of Coefficients, $c_{,}$

1. Initialize all coefficients. Set $c_{mj} \leftarrow 0$.

2. Select the first root. Let $j \leftarrow 1$.

3. Determine the proper value of b_j. Let $b_j \leftarrow 1$.

4. Let $b_j \leftarrow b_j + 1$. If $cos(2\pi j/m) > 0.5/(b_j + 1)$ then *Step*4 else *Step*5.

5. Modify the coefficients. First, let $k \leftarrow 0$.

6. For $l \leftarrow 0$ to k let $c_{ml} \leftarrow c_{mk} + g_j\binom{k}{l} b_j^{k-l}/(b_j + 1 - r_j)^{k+1}$.

7. If the absolute value of $1/(b_j + 1 - r_j)^k$ is less than an acceptable error then proceed to *Step*8 else let $k \leftarrow k + 1$ and proceed to *Step*6.

8. Select the next root. Let $j \leftarrow j + 1$. If $j \leq (m - 1)$ then *Stop* else goto *Step*3.

For values of $m < 6$ the value of b is zero. Thus, the algorithm is simplified under this constraint since the sum in step 6 invokes only the term when $k = l$.

References

|All78| Arnold Allen. *Probability, Statistics and Queueing Theory.* Academic Press, New York, New York, 1978.

|BM85| F. Baccelli and A. Makowski. Simple computable bounds for the fork-join queue. *Proc. Conf. Inform. Sci. Systems*, 1985.

|BMT87| F. Baccelli, W. Massey, and D. Towsley. Acyclic fork-join queueing networks. *submitted to JACM*, 1987.

|Fli84| L. Flatto and S Hahn. Two parallel queues created by arrivals with two demands. *Siam J. Appl. Math.*, 44, 1984.

|Han75| P. Brinch Hansen. The programming language concurrent pascal. *IEEE Transaction on Software Engineering.*, 1, 1975.

|Hoa85| C.A.R. Hoare. *Communicating Sequential Processes.* Prentice-Hall International, London, 1985.

|KC67| Leonard Kleinrock and E.G. Coffman. Distribution of attained service in time-shared systems. *Journal of Computer System Science*, 1, 1967.

[KMR71] Leonard Kleinrock, R. Muntz, and Rodemich. The processor-sharing queueing models for time-shared systems with bulk arrivals. *Networks*, 1, 1971.

[NT85] R. Nelson and A.N. Tantawi. Approximate analysis of fork/join synchronization in parallel queues. *IBM Report RC11481*, 1985. to appear in IEEE Transactions on Computers.

[NTTS87] R. Nelson, D. Towsley, and A. Tantawi. Performance analysis of parallel processing systems. *to be presented at SIGMETRICS '87*, 1987.

[Pyl81] I.C. Pyle. *The Ada Programming Language*. Prentice-Hall International, London, 1981.

[Rom87] C. Gary Rommel. *Distributed Programs*. PhD thesis, University of Massachusetts, 1987. in preparation.

[TY86] D. Towsley and S. P. Yu. Bounds for two server fork-join queueing systems. *submitted to Operations Research*, 1986.

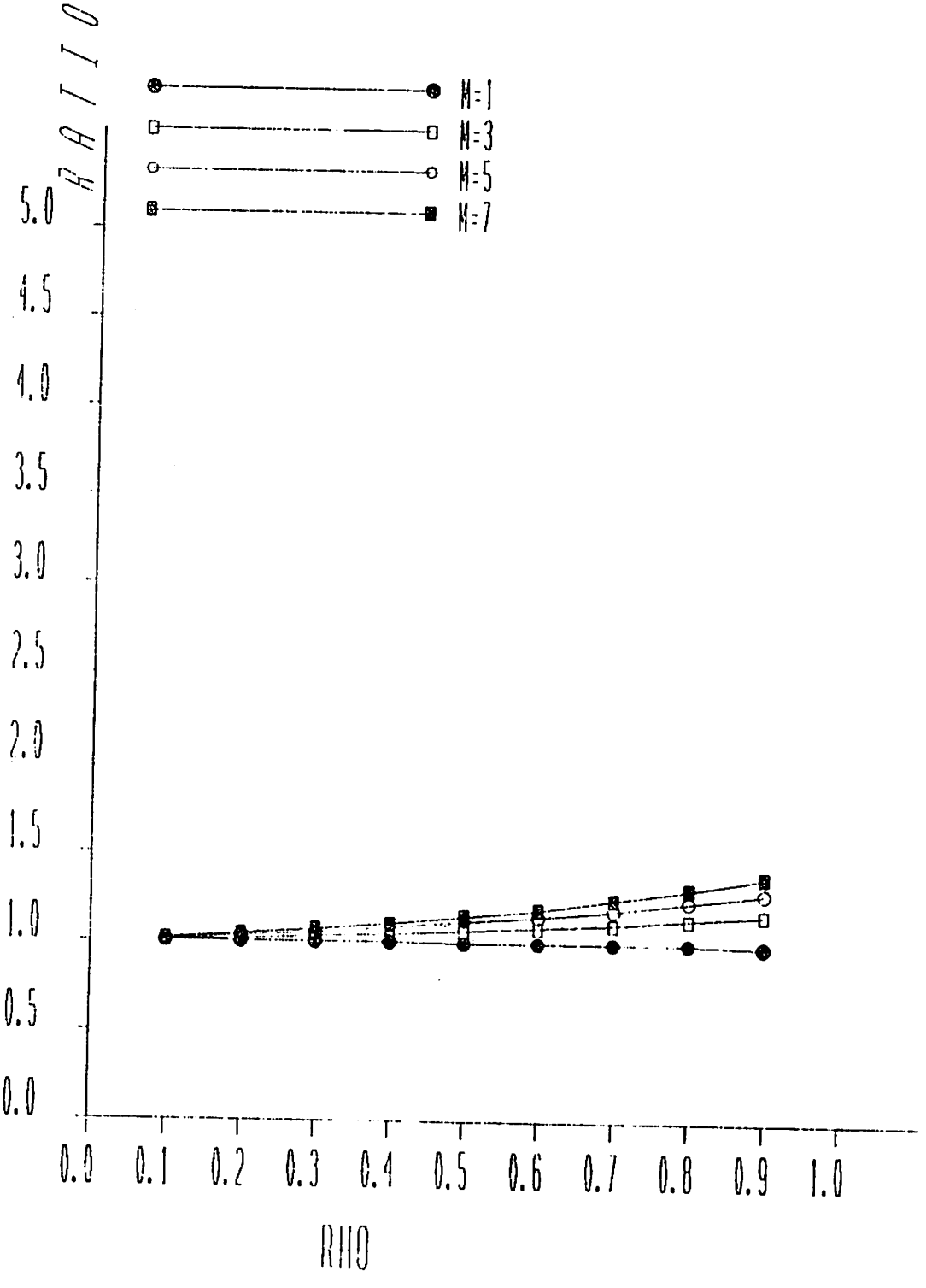

FIGURE 2 JOB/TASK

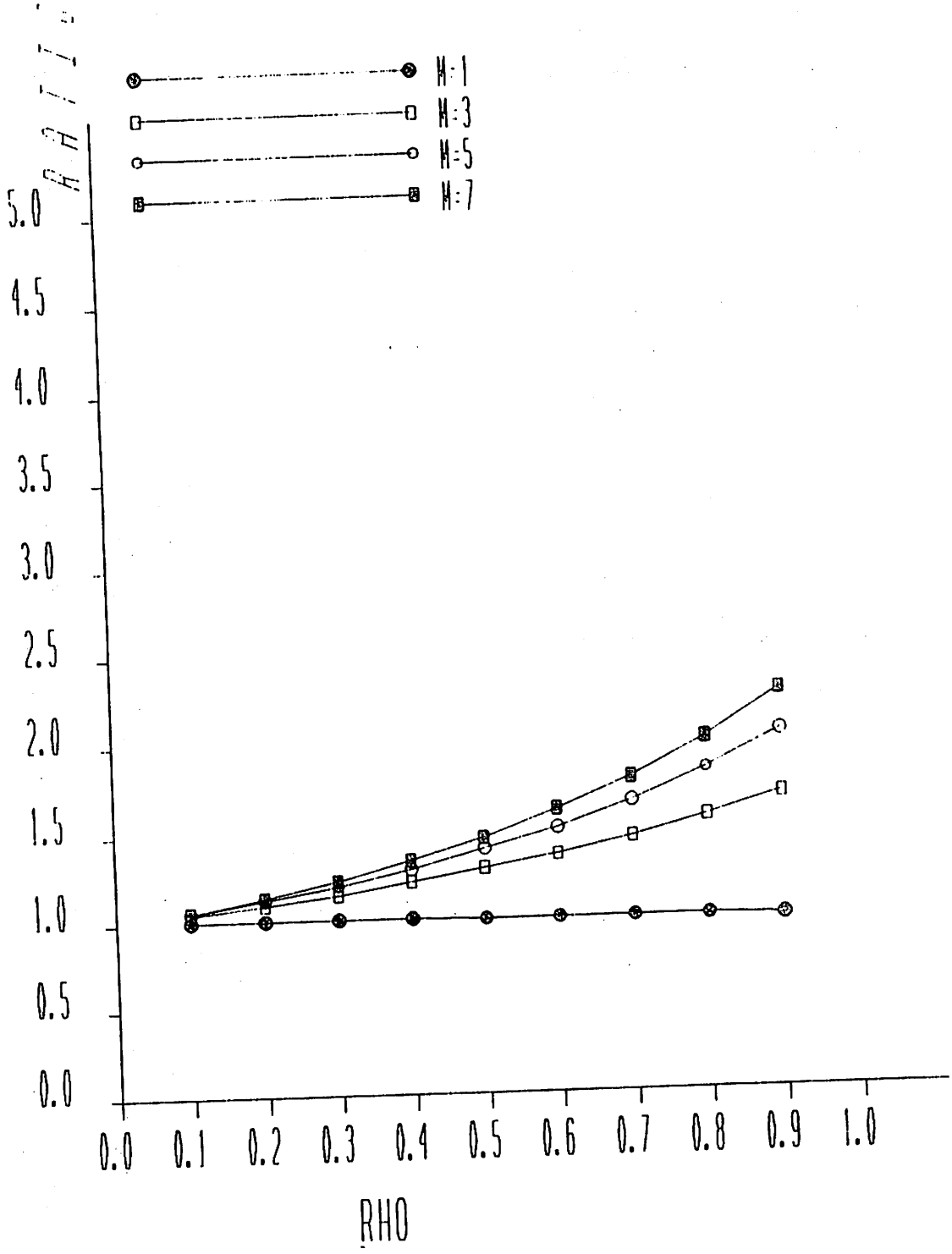

FIGURE 3 FCFS/TASK

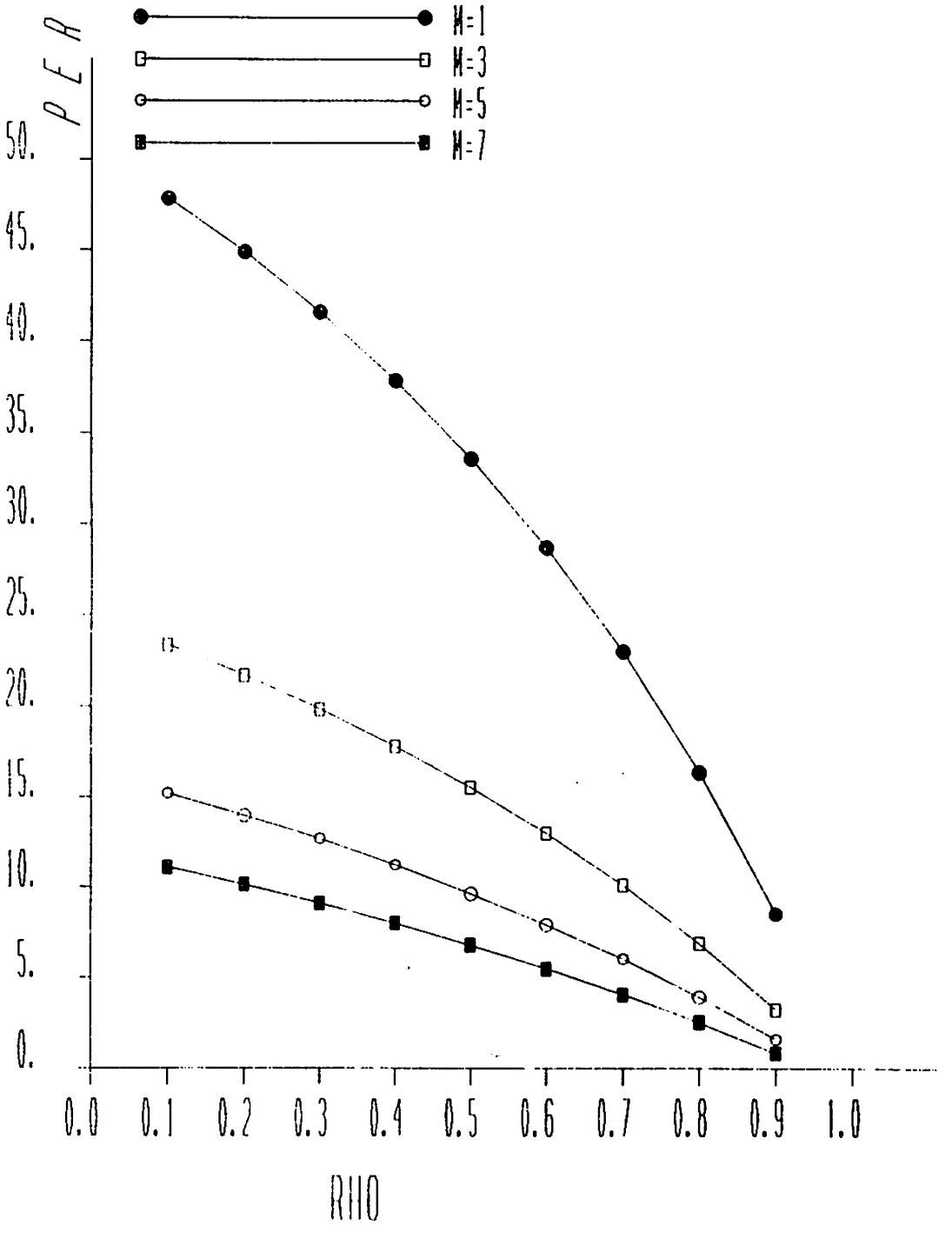

FIGURE 4A LOWER BOUND

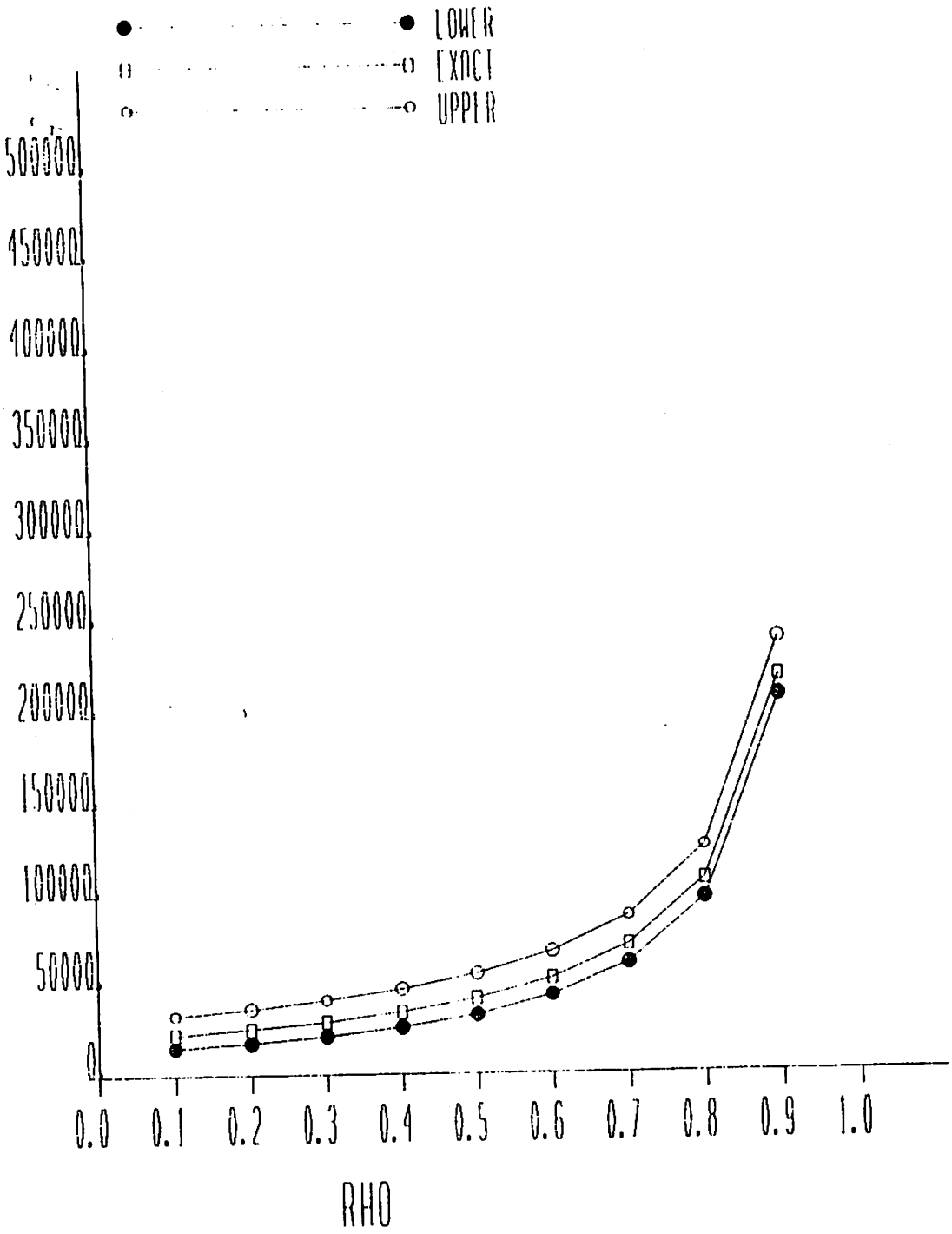

FIGURE 4B GROUP RESPONSE M=2

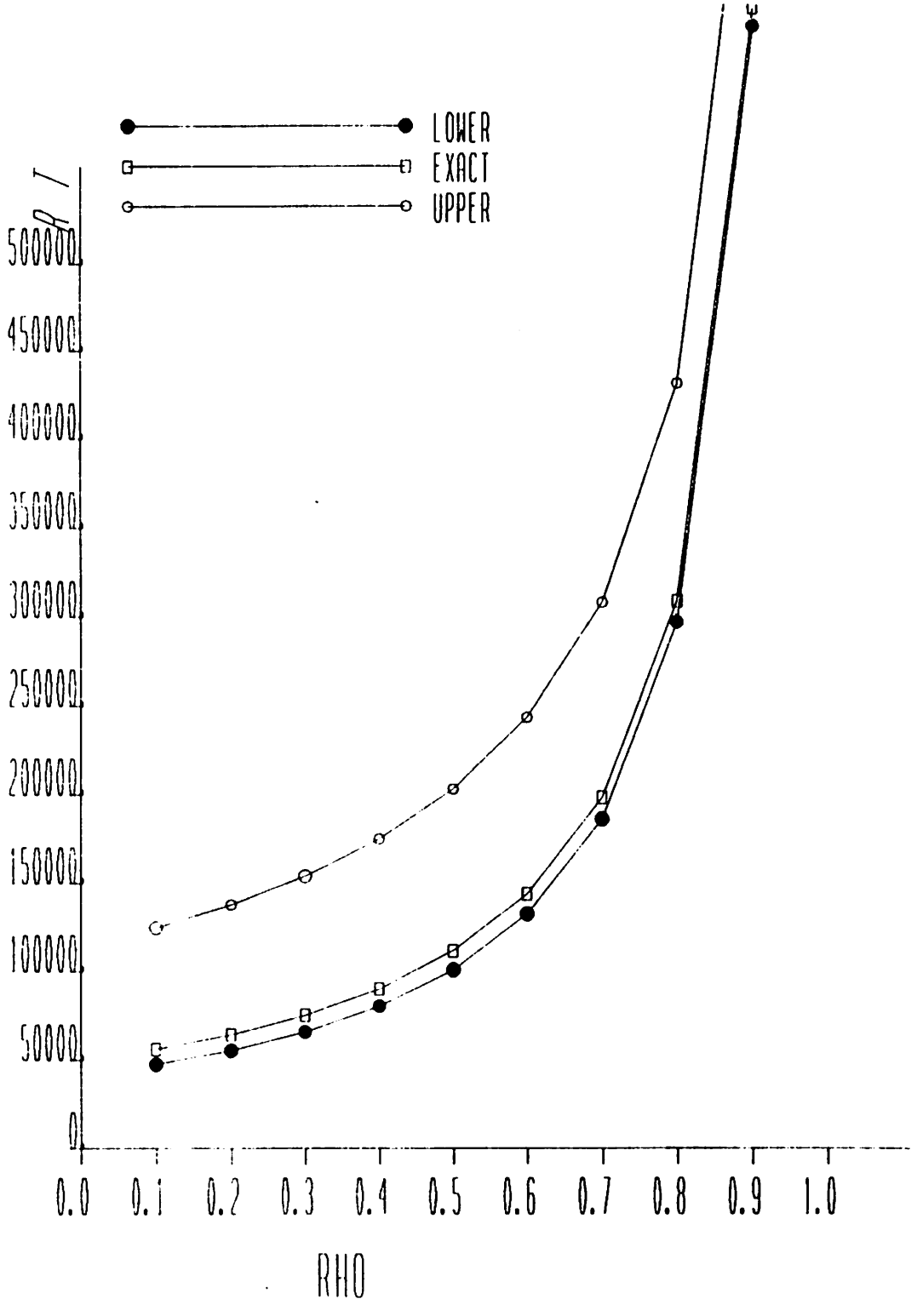

FIGURE 4C GROUP RESPONSE M-5

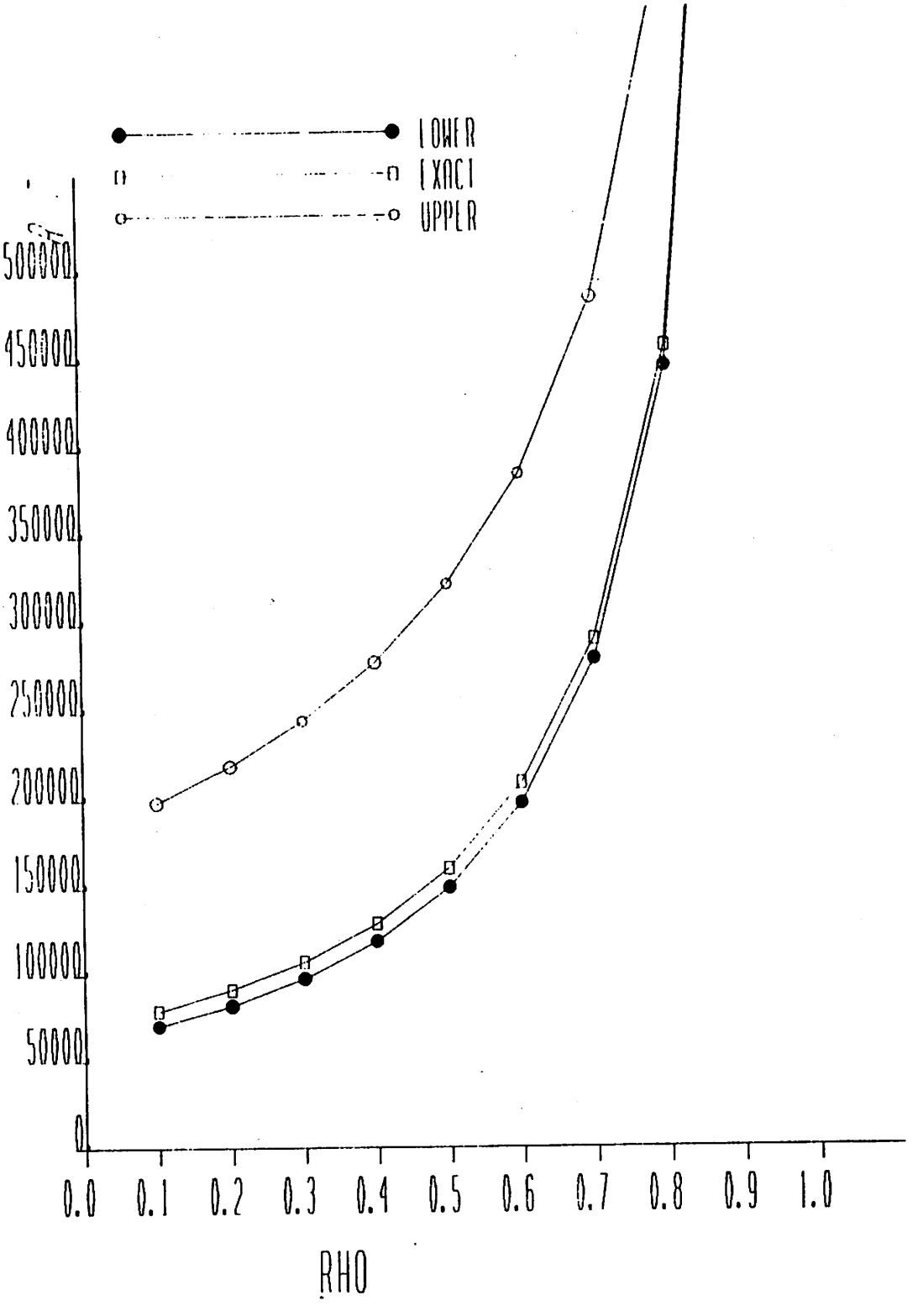

FIGURE 4D GROUP RESPONSE M=7

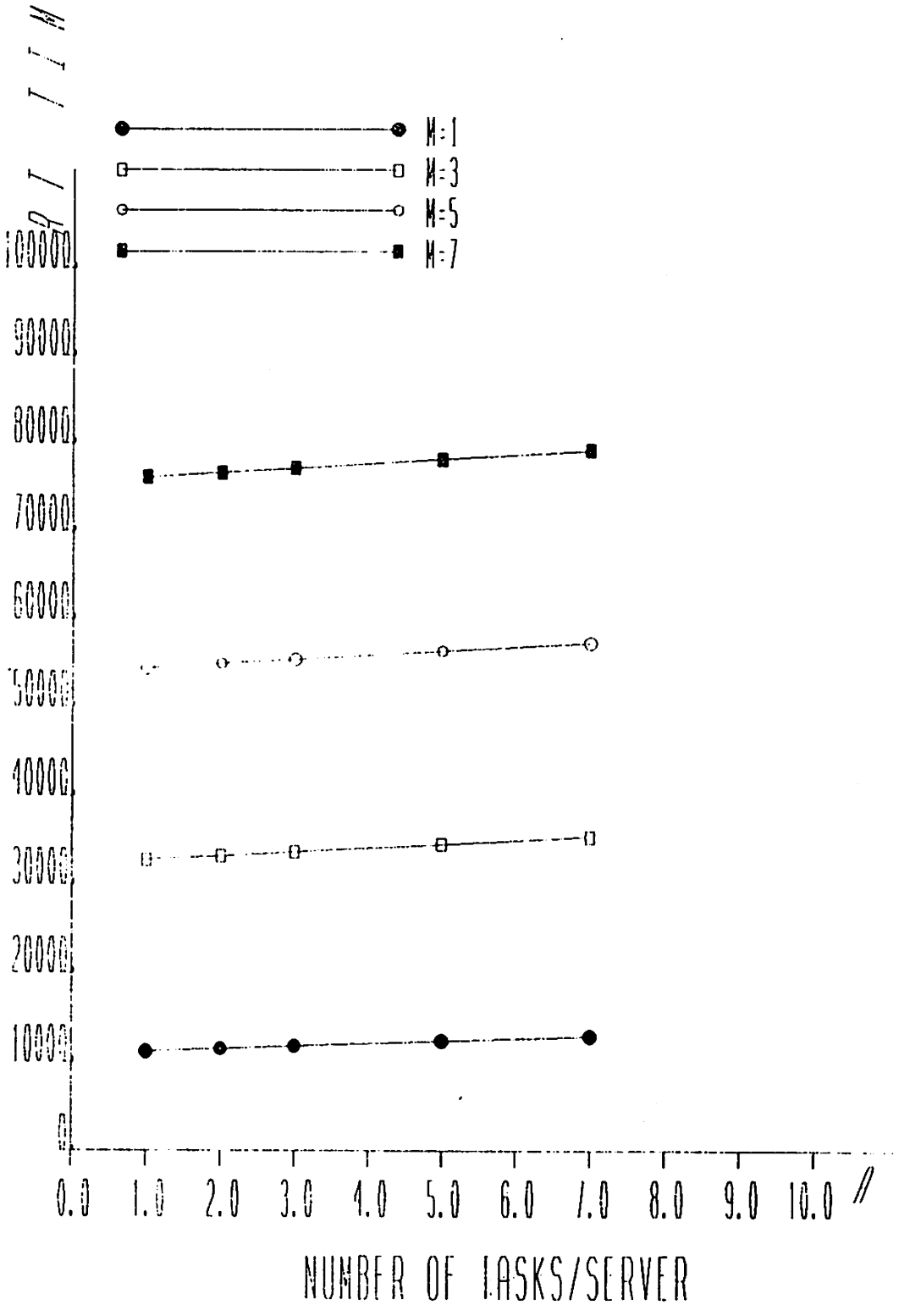

FIGURE 50 RESPONSE TIME RHO=.1

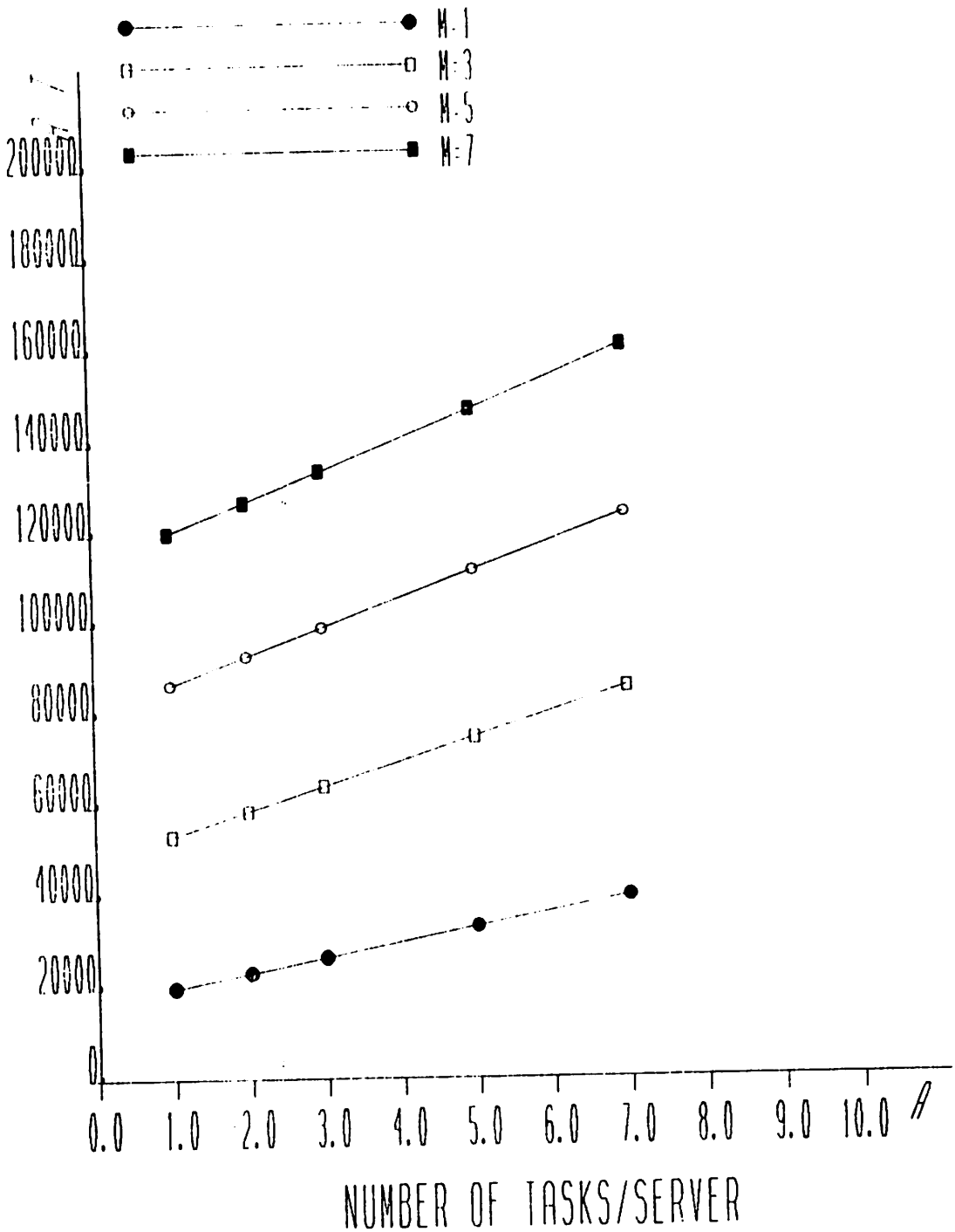

FIGURE 5B RESPONSE TIME RHO=.5

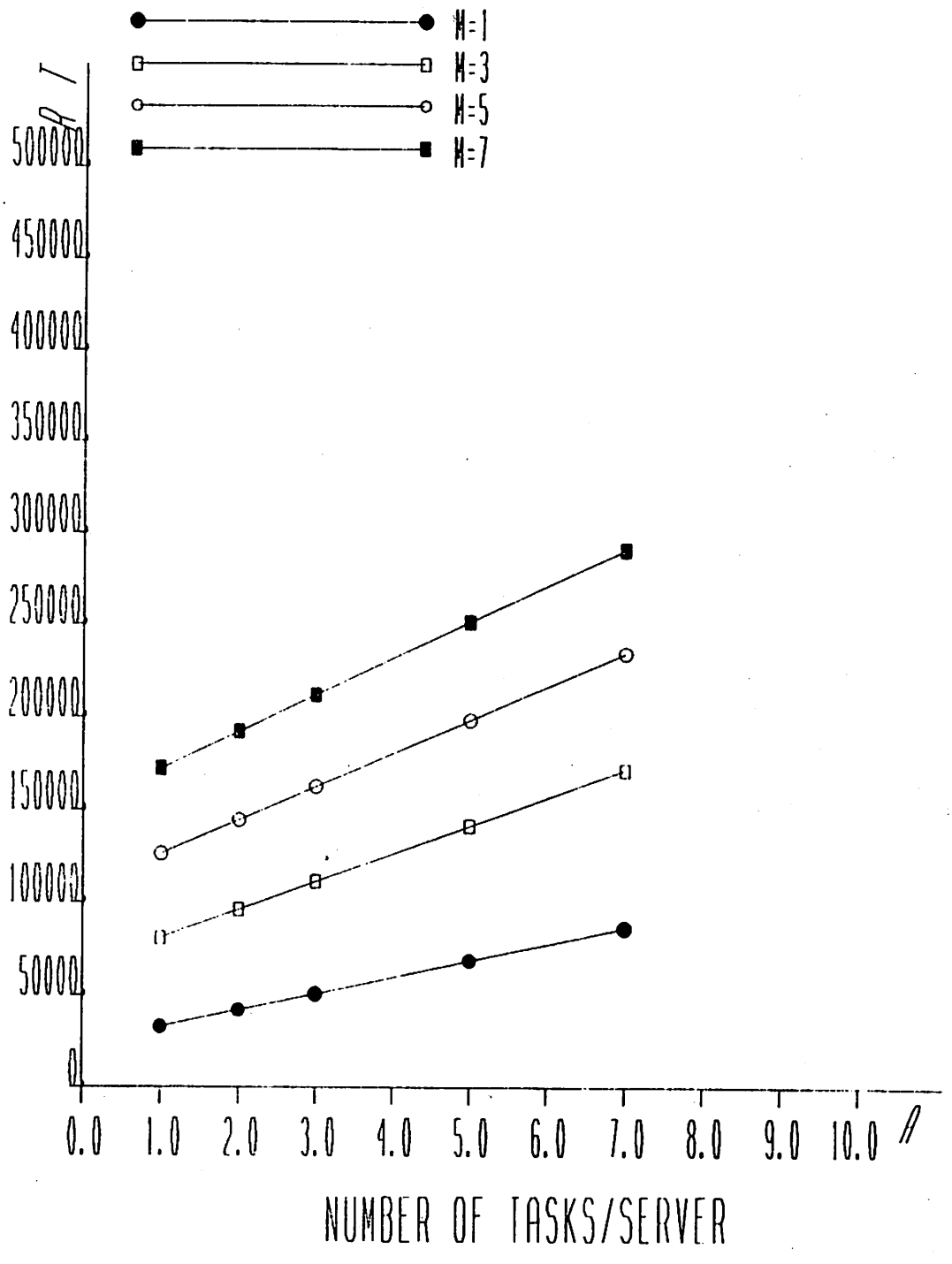

FIGURE 5C RESPONSE TIME RHO=.7

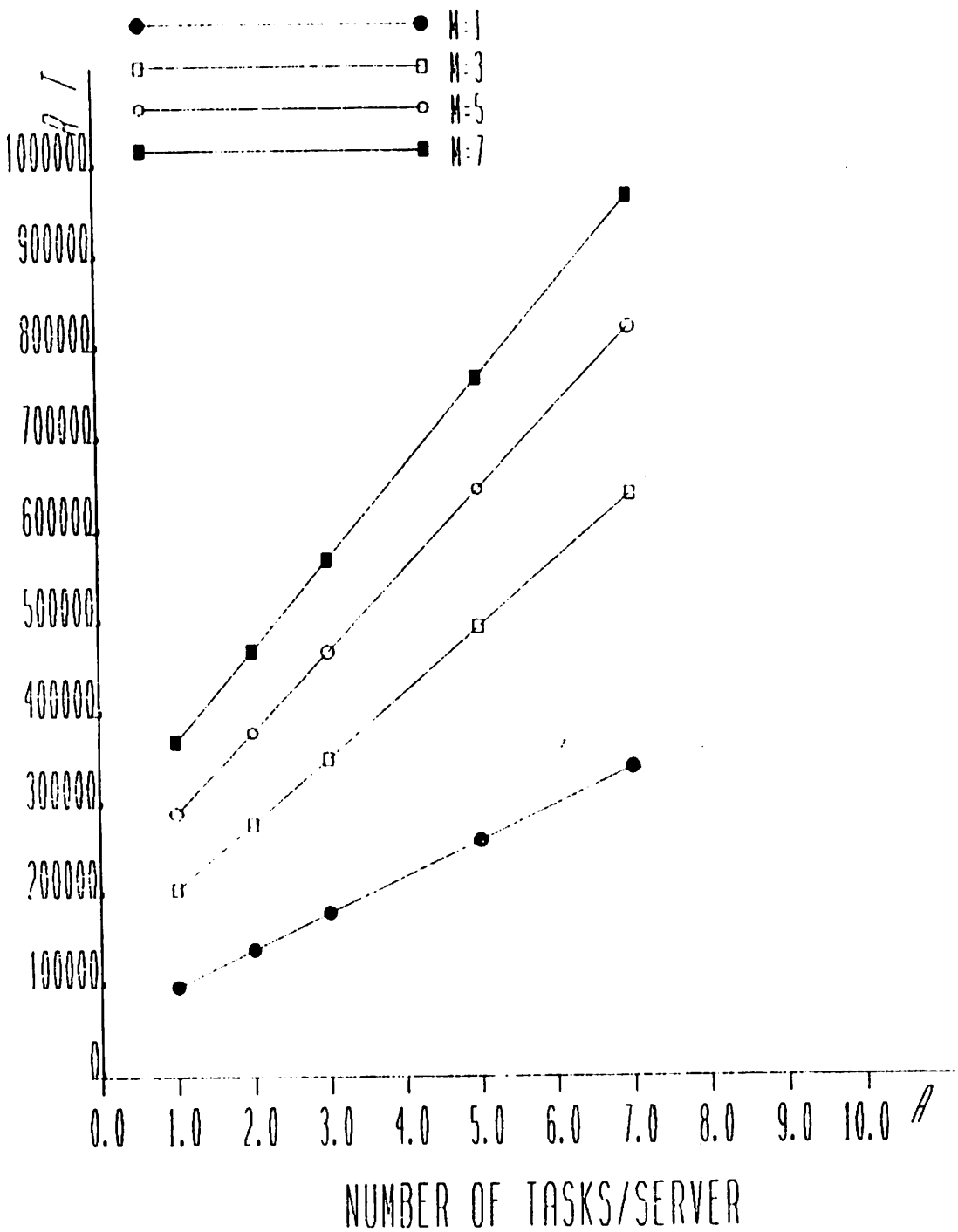

FIGURE 5D RESPONSE TIME RHO=.9

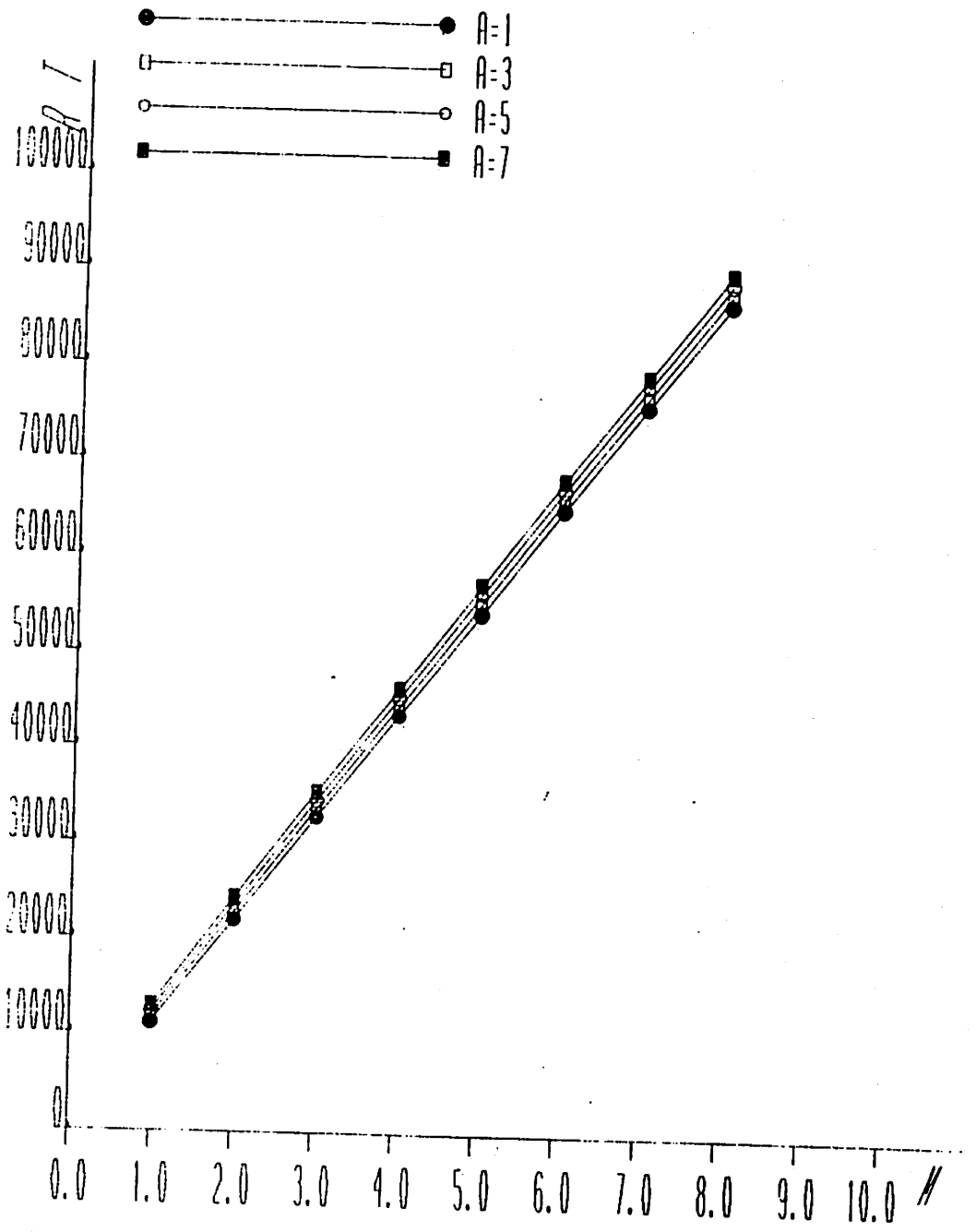

FIGURE 6A RESPONSE TIME RHO .1

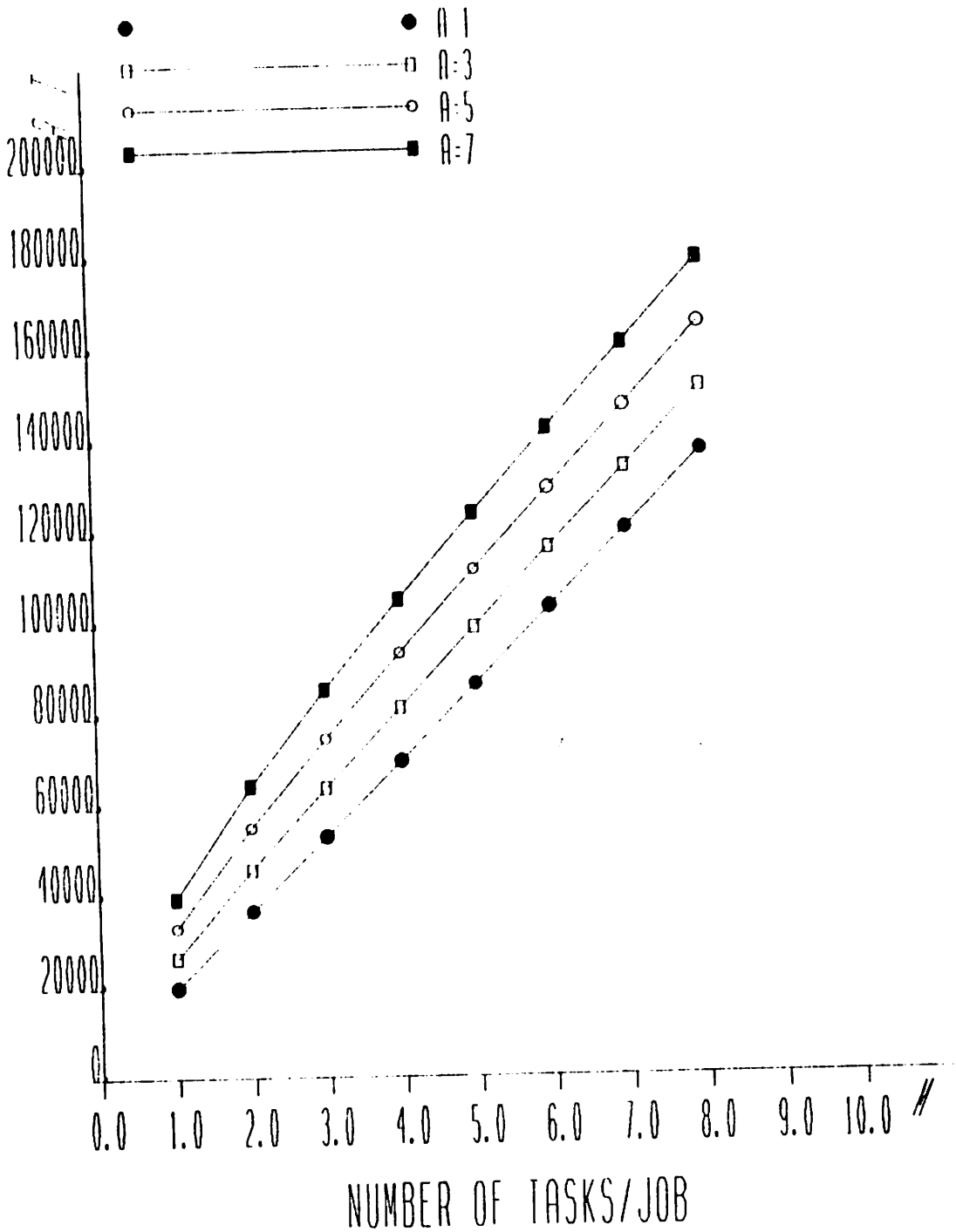

FIGURE 6B RESPONSE TIME RHO=.5

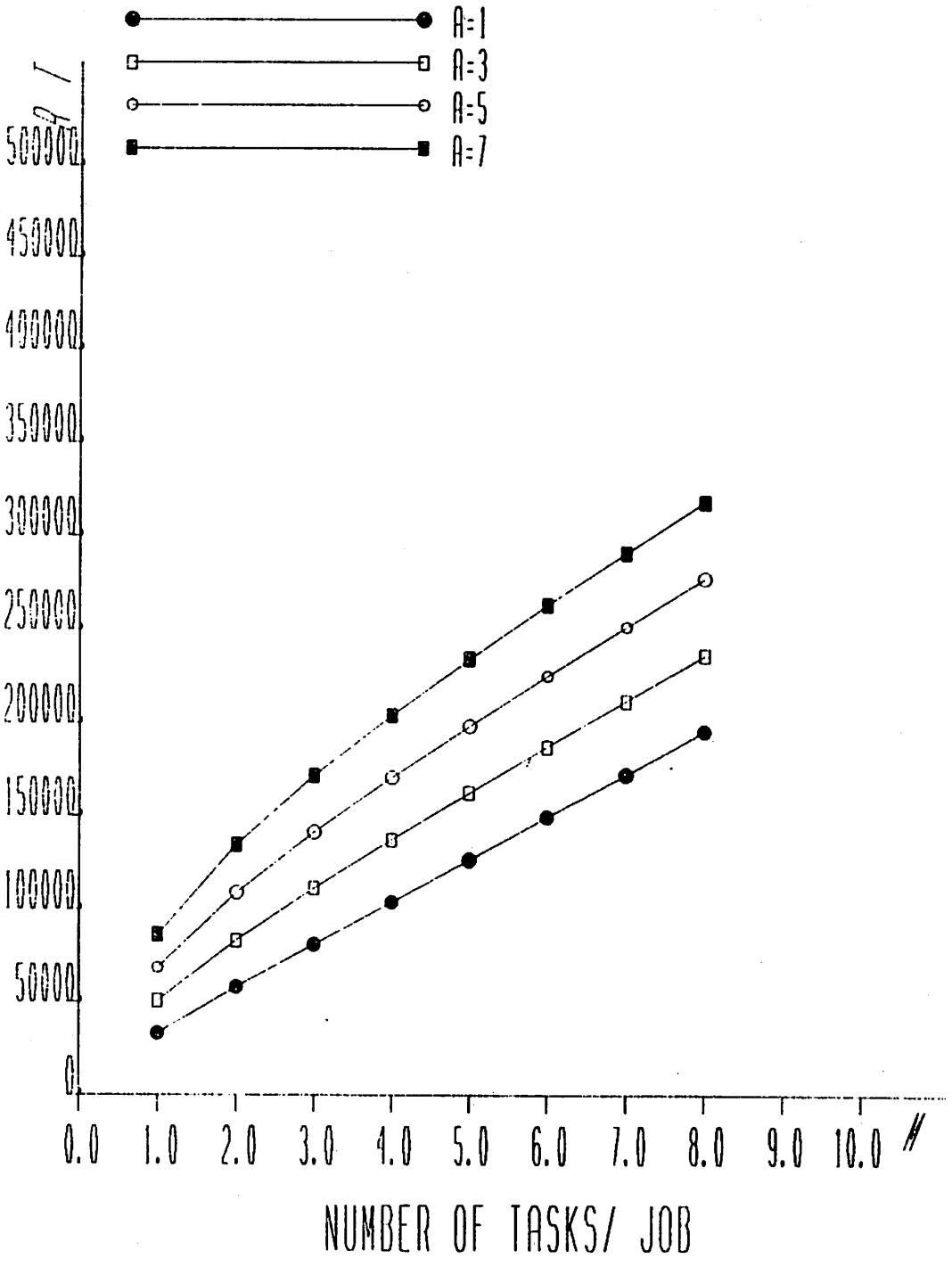

FIGURE 6C RESPONSE TIME RHO=.7

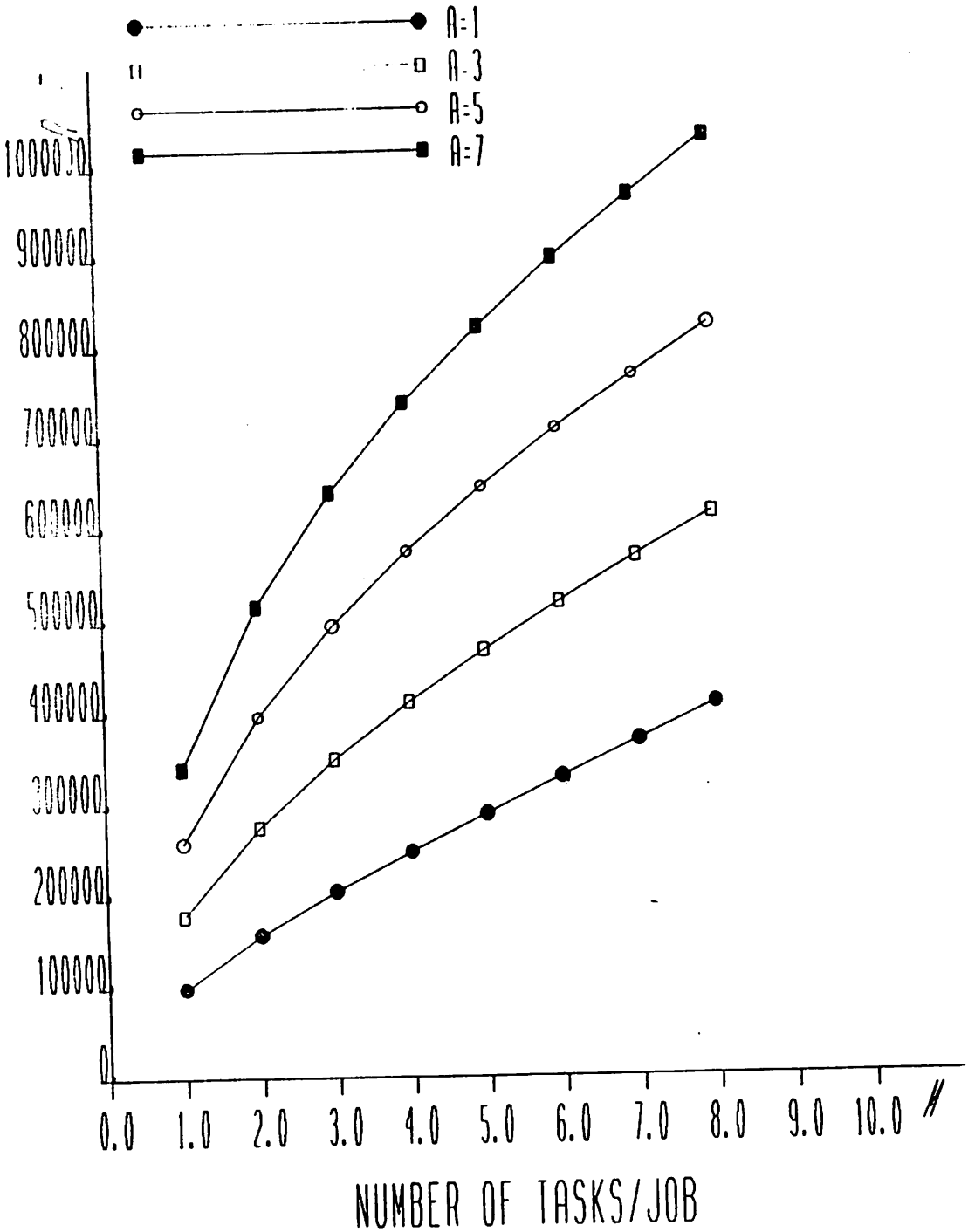

FIGURE 6D RESPONSE TIME RHO=.9