stichting
mathematisch
centrum

MC

AFDELING INFORMATICA IW 43/75 DECEMBER

C.L. PIPPEL & R. VAN VLIET

STOCHASTIC MODELS OF STORAGE ALLOCATION SYSTEMS

2e boerhaavestraat 49 amsterdam

Printed at the Mathematical Centre, 49, 2e Boerhaavestraat, Amsterdam.

The Mathematical Centre, founded the 11-th of February 1946, is a non-profit institution aiming at the promotion of pure mathematics and its applications. It is sponsored by the Netherlands Government through the Netherlands Organization for the Advancement of Pure Research (Z.W.O), by the Municipality of Amsterdam, by the University of Amsterdam, by the Free University at Amsterdam, and by industries.

AMS(MOS) subject classification scheme (1970): A55

ACM-Computing Reviews-category: 4.6,4.35,3.72,3.73

Stochastic models of storage allocation systems

by

C.L. Pippel & R. van Vliet

ABSTRACT

This paper presents a stochastic model of dynamic storage allocation systems. As an illustration we study an avail list system with and without garbage collection.

The mean lifetime of the system and the mean time between two subsequent occurrences of garbage collection are computed.

These values are only of limited practical use. The actually observed behaviour of dynamic storage allocation systems may considerably differ from the theoretical behaviour. This is due to the fact that the variance of the computed quantities is large.

KEY WORDS & PHRASES: *stochastic model, Markov chain, storage allocation, avail list, garbage collection.*

CONTENTS

1. INTRODUCTION

Stochastic techniques are widely used to study the behaviour of operating systems (c.f. [1], [2]). Another approach is simulation of important system components. We apply both techniques on dynamic storage allocation systems.

In chapters 2 and 3 we introduce the principles of a stochastic model. In chapters 4 and 5 we study two particular systems in detail. Both stochastic and simulation techniques are used to derive important system quantities. From the simulation it became apparent that small changes in the load of the system drastically altered the behaviour of it. This effect is discussed in chapter 6.

2. A DYNAMIC STORAGE ALLOCATION MODEL

In this section we present a model (DSA) in which a number of dynamic storage allocation systems can be described.

2.1. Definitions

A DSA-system controls a pool of storage, shared by several users. It consists of

- a storage buffer, in which data can be accommodated
- two routines, called request and return routine (referred to as P, respectively R).

The buffer is divided into a number of units. Each unit is either occupied (i.e. some users may have access to that unit to store and read data) or available (no users may have access to that unit).

The state of a unit is changed from occupied into available by the return algorithm. The state of a unit is changed from available into occupied by the request routine. The activation of the request routine is termed a *request*. The activation of a return routine is termed a *return*.

We restrict ourselves by the following assumptions:

- The units are numbered from 1 to a; a is the total number of units. All

units are structured identically.

- The user always requests a consecutive number of units. The request routine P has the following structure. It consists of a sequence of routines $P_0, P_1, \ldots, P_\ell$ that, when activated, either succeed or fail. When P is activated it first activates P_0. If P_0 succeeds, it has allocated the desired number of units to the user. Otherwise P activates P_1, etc. When $P_{\ell-1}$ fails, the system is full.

Each unit that may be allocated to a user by algorithm P_i and not by P_{i-1} [if present] is said to belong to level i. In the sequel we indicate DSA-systems meeting these limitations as leveled-systems.

The user may request a certain amount of storage by activating the request routine. This is termed a *request*. If the desired number of units is available, then the desired number of units is allocated to the user, otherwise the system is termed *full*. A user return units by activating the return routine. This is termed *return*. When the system is full some particular state change will occur. We assume an algorithm P_ℓ in P, which is activated when $P_{\ell-1}$ fails, and will always succeed.

3. STOCHASTIC MODELS

Leveled systems will be described by a stochastic model. First we briefly review the theory of Markov chains and then we present this model.

3.1. Markov chains

Consider a system which can be in one of the states of a set $S = \{E_j\}_{j \in J}$. The system behaves discrete in time in an indeterministic manner. We represent the state E_i of the system at the time N by $E_i^{(N)}$.

A system is said to be *Markov* when $E_i^{(N+1)}$ occurs with a certain probability p_{ik} which only depends on $E_k^{(N)}$ (all $k \in J$).

$$p_{ik} = \Pr(E_i^{(N+1)} \mid E_k^{(N)})$$

These quantities are non-negative and satisfy

$$\sum_{i \in S} p_{ik} = 1 \qquad k \in S$$

We call such a system a *finite Markov chain* when the state space S is finite. We arrange the conditional probabilities p_{ik} in the form of a transition matrix [i refers to a row, k to a column]:

$$P = (p_{ij})i,j \in S \tag{3.1}$$

Because the behaviour of the system is indeterministic the state of the system at time N is a probability distribution:

$$u^{(N)} = (u_k^{(N)})_{k \in S}$$

Where $u_k^{(N)}$ is the probability the system is in state k at time N. The following relation holds between $u^{(N+1)}$ and $u^{(N)}$

$$u_i^{(N+1)} = \sum_{k \in S} p_{ik}\, u_k^{(N)} \qquad (\forall i \in S)$$

This set of equations can be written in the from of a matrix equation:

$$u^{(N+1)} = P\, u^{(N)}$$

As a consequence:

$$u^{(N+1)} = P^{N+1} \cdot u^{(0)}$$

Define $P_{ij}^{(N)}$ as the probability of moving from state E_j to the state E_i in N steps. From the previous equation it is obvious that the matrix $(P_{ij}^{(N)})$ is P^N.

We study the asymtotic behaviour of $u^{(N)}$:

THEOREM 3.1. *Suppose that all* $p_{ij} > 0$ *and consider*

$$u^{(N)} = P^N \cdot u^{(0)}$$

Then whatever the values of the elements of $u^{(0)}$ *there exists a unique vector* $u = (u_i)$ *such that each* $u_i > 0$ *and such that* $u^{(N)}$ *converges to* u

as N *increases. In fact:*

$$|p_{ik}^{(N)} - u_i| \leq |1 - b \cdot \delta|^N$$

where

$$\delta = \min_{i,j} p_{ij}$$

b *equals the number of states of the system.*

PROOF. (Cf. [4], theorem 4.1.) □

THEOREM 3.2. *Central limit theorem. If there exists an integer* R *such that* $p_{ij}^{(R)} > 0$ *for all* i,j, *i.e. after* R *steps every state is accessible from every other state, then there exists a vector* $u = (u_i)$ *for which each* $u_i > 0$ *and*

$$|p_{ij}^{(N)} - u_i| \leq (1-\delta_R)^s \qquad \text{(all N)}$$

where

$$\delta_R = \min_{i,j} p_{ij}^{(R)}$$

and s *is the integral part of* $N \cdot R^{-1}$. *Moreover*

$$u = P \cdot u \tag{3.2}$$

PROOF. (Cf. [4], theorem 4.2.) □

REWORK. Markov chains, mentioned in the previous theorem are referred to as *ergodic* Markov chains. u is said to be the *stationary distribution* of the Markov chain. In the sequel we restrict ourselves to finite ergodic Markov chains.

Instead of (3.2) we may write

$$M \cdot u = 0 \tag{3.3}$$

where

$$M = P - I$$

M is the *stochastic matrix* of the Markov chain.

Given a Markov chain with a state space S, a transition matrix P and

a stationary distribution u. We derive a *sub Markov* chain as follows:

Let

$$S = S_1 \cup S_2 \cup \ldots \cup S_n$$

and

$$S_i \cap S_j = 0 \qquad \forall_{i,j}.$$

We define the state space of the subsystem as

$$\tau = (S_1, S_2, \ldots, S_n).$$

We define the transition matrix Q as

$$Q_{AB} = \sum_{k \in A} \sum_{i \in B} u_i \, u_B^{-1} \, p_{ki} \tag{3.4}$$

where

$$u_B = \sum_{i \in B} u_i$$

THEOREM 3.3. *There exists a stationary distribution for the subsystem. In particular:*

$$u = (u_A)_{A \in \tau}$$

PROOF. We have to prove:

$$Q \cdot u = u \tag{3.5}$$

Indeed:

$$\begin{aligned} \sum_B Q_{AB} \, u_B &= \sum_B \sum_{k \in A} \sum_{i \in B} u_i \, u_B^{-1} \, p_{ki} \cdot u_B \\ &= \sum_{k \in A} \sum_{i \in S} u_i \, p_{ki} \\ &= \sum_{k \in A} u_k \\ &= u_A \qquad \square \end{aligned}$$

We will use this theorem to prove properties about complicated Markov chains by studying simpler ones.

3.2. The microscopic model

3.2.1. *The state space*

The state of a leveled system can be characterized by specifying to which level each of the units belongs. Therefore

let $B = \{i \mid 1 \le i \le a\}$

let ℓ be the maximum level

let $L = \{k \mid 0 \le k \le \ell\}$

let $S = \{f : B \to L\} = L^B$

We establish a [1-1] correspondence between S and the set of possible states of the system as follows: To a given state of the system a function $f \in S$ corresponds, such that $f(i)$ is the level number of the i^{th} element of the buffer.

3.2.2. *Actions*

Request and return are termed *actions*. When an action occurs the state of the system changes. A request may be characterized by an integer $1 \le i \le a$ indicating the number of units requested.

Return may be characterized by i and b indicating that the units b, b + 1 ... b + i + 1 are returned.

Let f be the current state of the system. $\Pi_f(i)$ denotes the probability that the next action will be a request of i units. $\rho_f(i,b)$ denotes the probability that the next action will be the return of the units b, b + 1 ... b + i - 1. $\Pi_f(i)$ and $\rho_f(i,b)$ satisfy the following conditions:

(3.1) $\quad -\Pi_f(i) \ge 0 \qquad \forall i,f$

(3.2) $\quad -\rho_f(i,b) \ge 0 \qquad \forall i,b,f$

(3.3) $\quad -\sum_{i=1}^{a} \Pi_f(i) + \sum_{f=1}^{a} \sum_{i=1}^{a} \rho_f(i,b) = 1 \qquad \forall f$

3.2.3. *The system*

An action changes the state of the system. State transitions may be represented by the following matrices:

[we assume $f_1, f_2 \in S$]

- $P_{f_1 f_2}(i) = 1$ if a request of i units changes the state from f_2 to f_1.
  0 otherwise.
- $R_{f_1 f_2}(i,b) = 1$ if a return of i units starting at b changes the state from f_2 into f_1.
  0 otherwise.

As mentioned before P consists of a sequence of routines $\{P_0, P_1, \ldots, P_\ell\}$. At each request the routines are activated sequentially as necessary. Thus either P_0 is activated only or P_0, P_1 are activated or $P_0, P_1, \ldots, P_v$ $(0 \leq v \leq \ell)$. The routines $P_0, P_1, \ldots$ usually require an increasing amount of time. Which is the highest routine activated in a given state is a question of great importance. Therefore we introduce additional matrices:

DEFINITION.

$V_{vf}(i) = 1$ if a request in state f of i units activates the routines $P_0 \ldots P_v$ and not P_{v+1}. [This will be called a *recovery of type v*.]

As a consequence of the algorithm P being completely deterministic:

$$\sum_{v=0}^{\ell} V_{vf}(i) = 1.$$

DEFINITION. The *recovery matrix* Y.

Let Y_{vf} be the probability that in state f a recovery of type v occurs at the next transition.

$$Y_{vf} = \sum_{i=1}^{a} \Pi_f(i) V_{vf}(i)$$

For ergodic systems we add:

DEFINITION. *Stationary distribution* u

$$u = \{u_f\}_{f \in S}$$

DEFINITION. Y_v

Y_v is the probability that at the next transition a recovery of type v

occurs:

$$Y_v = \sum_{f \in S} u_f \, Y_{vf}$$

Finally we compute the transition matrix W

$$W_{f_1 f_2} = \Pr\{E_n = f_1 \mid E_{n-1} = f_2\}$$

$$W_{f_1 f_2} = \sum_{i=1}^{a} \Pi_{f_2}(i) \, P_{f_1 f_2}(i) + \sum_{i=1}^{a} \sum_{b=1}^{a} \rho_{f_2}(i,b) \, R_{f_1 f_2}(i,b)$$

From (3.1), (3.2), (3.3) we derive:

$$\sum_{f_1 \in S} W_{f_1 f_2} = 1 \qquad \forall f_2 \in S$$

3.3. The macroscopic model

In the previous section we described in detail the principles of our model. It should be clear that this model is not suitable for the following reasons:

- even in small systems the number of states is very large $[(\ell+1)^a]$. This makes practical calculations cumbersome or even impossible.
- The probability functions Π and ρ are difficult to estimate.

Moreover, we are mainly interested in the quantities Y_v, since these quantities indicate the practical performance of the system. As described in 3.1 we derive from the microscopic system a macroscopic system by taking several states together. The state f of the new system can be represented by a sequence of ℓ numbers:

$$f' = [f_0, f_1, \ldots, f_{\ell-1}]$$

f' corresponds to a set H_f of old states. Each element of this set has f_0 units in level 0, f_1 units in level 1, etc. We have to change all other quantities accordingly:

$$-\Pi'_{f}(i) = \sum_{f \in H_f} \Pi_{f}(i)$$

$$-\rho'_{f'}(i) = \sum_{f' \in H_f} \sum_{b=1}^{a} \rho_{f'}(i,b)$$

Note that we omit the second parameter of ρ' since the exact position of the returned units in the buffer is no longer of any interest for our system.

The new matrices P', R', V' must be constructed from the algorithms P and R. This is usually done by heuristic techniques, which we will not describe here. As mentioned before we define

- the transition matrix

$$W'_{f'_1 f'_2} = \sum_{i=1}^{a} \Pi_{f'_1}(i)\, P'_{f'_1 f'_2}(i) + \sum_{i=1}^{a} \rho_{f'_2}(i)\, R'_{f'_1 f'_2}(i)$$

- the recovery matrix

$$Y'_{vf'} = \sum_{i=1}^{a} \Pi_{f'}(i)\, V'_{vf'}(i)$$

- the stationary distribution

$$u'_{f'} = \sum_{f \in H_{f'}} u_f$$

-

$$Y'_{v} = \sum_{f'} u'_{f'}\, Y'_{vf'}$$

In the sequel we will omit the primes.

4. THE AVAIL LIST SYSTEM

4.1. Introduction

As a first example we treat an avail list system [al-system]. An al-system has a *buffer* of a units numbered from 1 up to and including a. Each unit is either free or occupied. In such a system we have only two levels.

The units in level zero are termed *free*. The units in level one are termed *occupied*. According to chapter (3.3) we indicate the state of the system by:

$$[i] \qquad 0 \leq i \leq a$$

Usually the user will not bother about the state of the buffer doing requests and returns as he likes. Of course, when the number of occupied units becomes small, the probability that the next action will be a return diminishes. In general we may assume that $\Pi_f(i)$ and $\rho_f(i)$ are independent of f_1 but some kind of reflection is involved at the bound of an empty buffer. When the system is large the exact form of the boundary condition does not seriously influence the behaviour of the system. As a further simplification we assume that only one unit is requested or returned at a time. Hence:

$$\Pi_f(i) = p \qquad i = 1$$

$$\Pi_f(i) = 0 \qquad \text{otherwise}$$

$$\rho_f(i) = r \qquad i = 1$$

$$\rho_f(i) = 0 \qquad \text{otherwise}$$

$$p + r = 1.$$

Note:

The boundary condition for a return when the buffer is empty will be brought in by assuming that the return routine leaves the state of the buffer unchanged in this situation. This means that in state [0] there is a probability r that at the next action "nothing" will happen. This assumption violates our definition of action.

When the system is full the system changes to [i] with a probability δ_i.

$$\sum_{i=0}^{a} \delta_i = 1 \qquad 0 \leq i \leq a$$

This completes our model of al-systems.

For future use we introduce

$$z = r \cdot p^{-1}.$$

Intuitively one would expect z = 1. For over a long period z equals:

the number of returns / the number of requests.

However, the number of returns is not necessarily equal to the units actually returned. On the one hand we allow a return when the buffer is empty without actually returning a unit and on the other hand extra units are returned when the buffer is full. [In the latter case information moved to some backing storage.]

4.2. The stationary distribution

Fig. (4.1) diagrams the state transitions involved in an al-system.

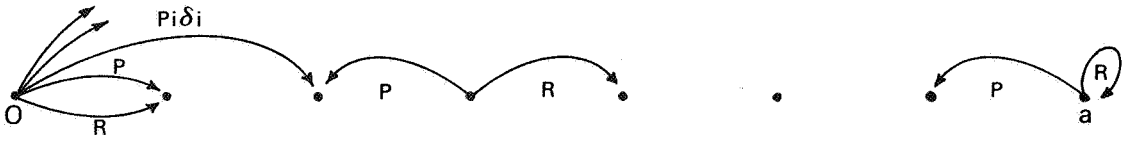


Fig. 4.1

We indicate the stationary distribution of an al-system by

$$u = \{u_i\}_{i=0}^{a}$$

where u_i is the probability of being in state [i]. The stationary distribution $u = \{u_i\}_{i=0}^{a}$ is a solution of:

$$\begin{aligned} &(p+r)\ u_i = r\ u_{i-1} + p\ u_{i+1} + \delta_i\ p\ u_0 \\ &(p+r)\ u_a = r\ u_{a-1} + r\ u_a + \delta_a\ p\ u_0 \\ &(p+r)\ u_0 = p\ u_i + p\ \delta_0 u_0 \end{aligned} \qquad (4.1)$$

This yields the following stochastic matrix:

$$M = \begin{pmatrix} p\delta_0 & -(p+r) & p & & & & \\ p\delta_1 + r & -(p+r) & p & & & & \\ & r & -(p+r) & p & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & r & -(p+r) & p & \\ & & & & r & -(p+r) & p \\ & & & & & r & -p \end{pmatrix}$$

$$(4.2) \qquad M.u = 0$$

As the equation is homogeneous and we know that a one-dimensional nontrivial solution exists [the system being ergodic], we may assume

$$u_0' = 1$$

Of course we have to normalize later.

Let

$$W_k = \sum_{i=0}^{k} z^i \qquad k \geq 0$$

$$W_k = 0 \qquad k \leq 0$$

LEMMA.

$$u_k' = W_k - \sum_{i=0}^{k-1} \delta_i \, W_{k-1-i}$$

PROOF. (by induction to k)

$$k = 0 \qquad u_0' = 1 = W_0$$

Assume that we have verified this solution for $u_0', \ldots, u_k'$. Observe that

$$(4.3) \qquad W_{k+1} = (1+z)W_k - z^{-1} W_{k-1}$$

From the induction hypothesis and (4.3) we derive:

$$u'_{k+1} = (p+r).p^{-1}\, u'_k - r.p^{-1}\, u'_{k-1} - \delta_k\, u'_0$$

$$= (1+z)\left(W_k - \sum_{i=0}^{k-1} \delta_i\, W_{k-1-i}\right) - z^{-1}\left(W_{k-1} - \sum_{i=0}^{k-z} \delta_i\, W_{k-2-i}\right) - \delta_k\, W_0$$

$$= (1+z)W_k - z^{-1}\, W_{k-1}$$

$$- (1+z) \sum_{i=0}^{k-1} \delta_i\, W_{k-1-i} + z^{-1} \sum_{i=0}^{k-z} \delta_i\, W_{k-2-i} - \delta_k\, W_0$$

$$= W_{k+1} - \sum_{i=0}^{k-1} \delta_i\left((1+z)W_{k-1-i} - z^{-1}\, W_{k-2-i}\right) - \delta_k\, W_0$$

$$= W_{k+1} - \sum_{i=0}^{k} \delta_i\, W_{k-i} \qquad \square$$

These values of u'_i still must be normalized.

Let

$$s = \sum_{i=0}^{a} u'_i$$

The normalized solution of (4.1) is:

$$u_i = u'_i \,.\, s^{-1}$$

$$s = \sum_{k=0}^{a}\left(W_k - \sum_{i=0}^{k-1} \delta_i\, W_{k-1-i}\right)$$

$$= \sum_{k=0}^{a} W_k - \sum_{k=0}^{a}\sum_{i=0}^{k-1} \delta_i\, W_{k-1-i}$$

$$= \sum_{k=0}^{a} W_k - \sum_{i=0}^{a} \delta_i \sum_{k=0}^{a-1-i} W_k$$

$$= \sum_{i=0}^{a} \delta_i \left(\sum_{k=0}^{a} W_k - \sum_{k=0}^{a-1-i} W_k\right) \qquad \text{[using } \sum_{i=0}^{a} \delta_i = 1\text{]}$$

Let

$$t_i = \sum_{k=0}^{a} W_k - \sum_{k=0}^{a-1-i} W_k \tag{4.4}$$

Then

$$s = \sum_{i=0}^{a} \delta_i \, t_i \tag{4.5}$$

Two cases arise:

$z = 1$

$$t_i = \sum_{k=0}^{a} (k+1) - \sum_{k=0}^{a-1-i} (k+1)$$

$$= \tfrac{1}{2}(a+1)(a+2) - \tfrac{1}{2}(a-i)(a-i+1)$$

$$= \tfrac{1}{2}(i+1)(2a+2-i) \tag{4.6.1}$$

$z \neq 1$

$$t_i = \sum_{k=0}^{a} \frac{1 - z^{k+1}}{1 - z} - \sum_{k=0}^{a-1-i} \frac{1 - z^{k+1}}{1 - z}$$

$$= \frac{z^{a+2} - z^{a-i+1}}{(1-z)^2} + \frac{i + 1}{(1-z)} \tag{4.6.2}$$

Finally we arrive at:

$z = 1$

$$u_k = \left((k+1) - \sum_{i=0}^{k-1} \delta_i \, (k-i)\right) . \left(\sum_{i=0}^{a} \delta_i \, (i+1)(2a+2-i) . \, 0.5\right)^{-1}$$

(4.7)

$z \neq 1$

$$u_k = \left(\frac{1 - z^{k+1}}{1 - z} - \sum_{i=0}^{k-1} \delta_i \, \frac{1 - z^{k-i}}{1 - z}\right) \left(\sum_{i=0}^{a} \delta_i \, \frac{z^{a+2} - z^{a-i+1}}{1 - z^2} + \frac{i + 1}{1 - z}\right)^{-1}$$

where $u = \{u_k\}_{u=0}^{a}$ is a solution of (4.1).

4.3. The mean recurrence time full (τ_f)

To simpify the forms of this section we assume that

$$\left.\begin{aligned} \delta_i &= 0 \qquad i \neq d \\ \delta_d &= 1 \end{aligned}\right\} \; 0 \leq d \leq a \tag{4.8}$$

The quantity we are interested in is Y_1, i.e. the probability that the system will be full at the next action. Let τ_f be the mean recurrence time of full. τ_f equals the average number of actions between two consecutive times the system is full.

$$\tau_f = (p.u_0)^{-1}$$

according to (4.6), (4.7):

$$\tau_f = (d+1)(2a+2-d) \qquad z = 1$$

(4.9)

$$\tau_f = \left(\frac{z^{a+2} - z^{a+1-d}}{(1-z)^2} + \frac{d+1}{(1-z)}\right)\left(\frac{z+1}{z}\right)$$

d equals the number of units released when the system is full.

Table (8.1.4) quotes some values of τ_f.

τ_f is a function of d, z and a. In the next chapters we consider τ_f as a function of d, z and a respectively.

4.3.1. $\tau_f(z)$

Table (8.1.4) shows that $\tau_f(z)$ heavily depends on the value of z. The relative fluctuation in z_0 is:

$$\frac{z - z_0}{z_0}$$

The relative fluctuation in $\tau_f(z_0)$ is:

$$\frac{\tau_f(z) - \tau_f(z_0)}{\tau_f(z_0)}$$

Between these quantities the following relation holds approximately:

$$\frac{\tau_f(z) - \tau_f(z_0)}{\tau_f(z_0)} = \frac{\tau_f'(z_0)}{\tau_f(z_0)} \cdot z_0 \quad \frac{z - z_0}{z_0}$$

$$\tau_f'(z_0) = \left.\frac{d\tau_f(z)}{dz}\right]_{z_0}$$

Let

$$\alpha(z_0) = \frac{\tau_f'(z_0)}{\tau_f(z_0)}\, z_0$$

Observe that:

$$\tau_f(z) = \frac{z+1}{z} \cdot t_0(z)$$

$$\tau_f'(z) = \frac{z+1}{z}\, t_0'(z) - \frac{1}{z^2}\, t_0(z)$$

$$\frac{\tau_f'(z)}{\tau_f(z)} = \frac{t_0'(z)}{t_0(z)} - \frac{1}{z(z+1)}$$

Hence

$$\alpha(z_0) = \frac{t_0'(z)}{t_0(z)} \cdot z_0 - \frac{1}{(z_0+1)}$$

For $z_0 = 1$ this yields:

$$t_0(1) = \tfrac{1}{2}(d+1)(2a+2-d)$$

$$t_0'(1) = \frac{\left((a+2)z^{a+1} - (a+d+1)z^{a-d} - az^{a+2} + (a-d-1)^{a+1-d} + (d+1)(1-z)\right)}{(1-z)^3}$$

Using the theorem of 1'Hopital three times:

$$t_0'(1) = \left((a+2)(a+1)a - (a-d+1)(a-d)(a-d-1)\right) / 6$$

As a consequence:

$$\alpha(1) = \frac{(a+2)(a+1)a - (a-d+1)(a-d)(a-d-1)}{(d+1)(2a+2-d)} \cdot \frac{1}{3} - \frac{1}{2}$$

We consider some special cases:

$d = 0$ $\qquad \alpha(1) = \tfrac{1}{2}(a-1)$

$d = a$ $\qquad \alpha(1) = \tfrac{1}{2}a - \tfrac{1}{2}$

Fluctuations in z cause fluctuations in $\tau_f(z)$ amplified by a factor $\alpha(z)$. To clarify the implication we quote some results:

$$a = 100 \qquad d = 0 \qquad z = 1 \qquad \alpha(1) = 50$$

A fluctuation of 1% in z causes a fluctuation of 50% in $\tau_f(z)$

$$a = 100 \qquad d = 100 \qquad z = 1 \qquad \alpha(1) = 33$$

A fluctuation of 1% in z causes a fluctuation of 33% in $\tau_f(z)$.
[These results only indicate the order of magnitude; table (8.1.5) quotes some values.]

4.3.2. $\tau_f(d)$

When the buffer is full, d units are made available again. Of course, when this number increases the mean recurrence time of full will increase and will be maximal if d = a.

To illustrate the behaviour of $\tau_f(d)$ we solve the equation:

$$2\tau_f(d) = \tau_f(a)$$

Three cases arise:

$z << 1$ We omit all exponents of z

$$2(d+1) = a + 1$$

$$d = \tfrac{1}{2}(a-1)$$

$z = 1$

$$2(d+1)(2a+2-d) = (a+1)(a+2)$$

$$2d^2 - 2(2a+1)d + a^2 - a - 2 = \emptyset$$

$$d \approx \tfrac{1}{2}(2a+1) - \tfrac{1}{2}\sqrt{2}(a+1\tfrac{1}{2})$$

$$d \approx 0.3a$$

$z >> 1$ We only take into account the exponents

$$2z^{a+1-d} = z^{a+2}$$

$$2z^{-(d+1)} = 1$$

$$d \approx \frac{\ln 2}{\ln z} - 1$$

In the common case $z \approx 1$ these calculations are of practical importance. It shows that the recurrence time of full is already half of its maximum when only thirty percent of the buffer is empty.

5. SYSTEMS WITH GARBAGE COLLECTION

In the previous chapter we have treated the avail list system. All units returned to the system can be recognized immediately as available. All available units belong to one level. In many systems the units returned to the system are set aside for a while. They are recollected when need arises. The process of recollection is called *garbage collection* (gc). The available units in these systems can be recognized as belonging to different levels. The units of level zero can be recognized as available immediately. Units set aside for a while belong to higher levels. They are recollected by some process of garbage collection. As an example we treat the following system.

The units of the system are numbered from 1 up to and including a. Units belong to one of the following levels:

0	free	can be recognized as available immediately
1	semi-free	can be recognized as free after garbage collection
2	occupied	

We request the state of the buffer by a pair

[s,f]

where:

f denotes the number of units in level zero
s denotes the number of units in level one

The number of occupied units equals

a - (f+s)

The request routine P behaves as follows: P_0 searches for units in level zero. If the requested number of units is not present, P_1 is called. P_1 adds the units of level one to level zero. If the requested number of units is not present in level zero the system is full and P_2 is called. P_2 adds d units from level 2 to level zero.

$0 \leq d \leq a$

The return algorithm adds the returned units to level one. As in the

previous chapter we consider p.q as the probability a request respectively return occurs.

5.1. The stationary distribution

Fig. (5.1) diagrams the state transitions involved in the system:

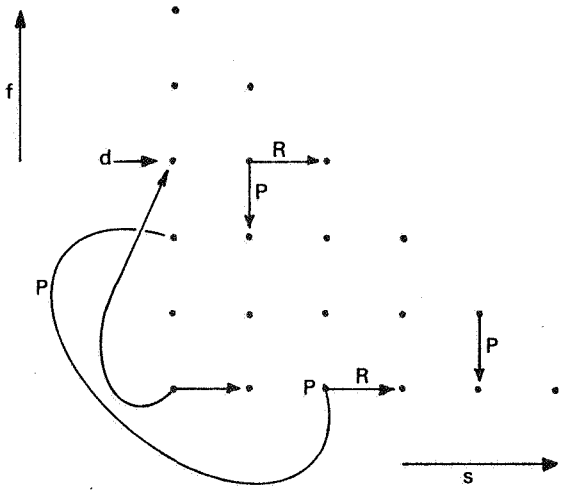


Fig. (5.1)

Let $\{u_{ik}\}_{i,k}$ denote the stationary distribution.

Let

$$0 \leq d \leq a$$

Let

$$\Delta_{di} = 1 \qquad d = i$$

$$\Delta_{di} = 0 \qquad \text{otherwise}$$

$\{u_{ik}\}$ satisfies the following equations:

(5.1.1) $$u_{ik} = r \cdot u_{i-1k} + p\, u_{i\,k+1} \qquad (0<i\leq a,\ 0\leq k<a)$$

(5.1.2) $$u_{i\,a-i} = r\, u_{i-1\,a-i} + (1-p)u_{i\,a-i} \qquad (0\leq i<a)$$

(5.1.3) $$u_{0i} = p\, u_{0i+1} + pu_{i+10} + \Delta_{di}\, pu_{00} \qquad (0\leq i<a)$$

(5.1.4) $$u_{0a} = (1-p)u_{0a} + \Delta_{da}\, pu_{00}$$

We arrange the components of the stationary distribution as depicted in fig. (5.2)

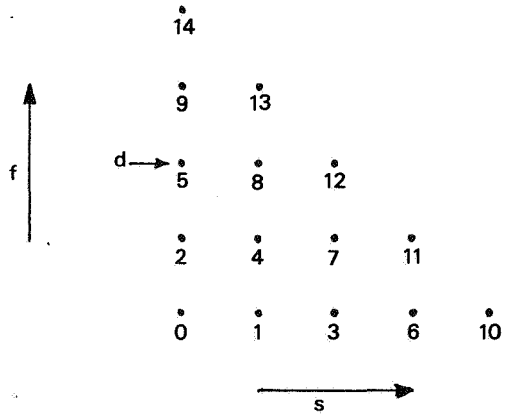


Fig. (5.2)

In fact we renumber the state of the system:

$$[i,k] \to [\tfrac{1}{2}(i+1)(i+k+1) + k]$$

The stochastic matrix M takes the following form:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	$-(p+r)$	p	p												
1	r	$-(p+r)$			p										
2			$-(p+r)$	p		p									
3		r		$-(p+r)$				p							
4			r		$-(p+r)$				p						
5	p					$-(p+r)$	p			p					
6				r			$-(p+r)$					p			
7					r			$-(p+r)$					p		
8						r			$-(p+r)$					p	
9										$-(p+r)$	p				p
10							r				$-p$				
11								r				$-p$			
12									r				$-p$		
13										r				$-p$	
14															$-p$

The stationary distribution u is a solution of

(5.2) $M \cdot u = 0.$

The quantities Y of the system are:

$$Y_{0(i,k)} = p \qquad i > 0$$
$$= 0 \qquad i = 0$$

$$Y_{1(i,k)} = 0 \qquad i > 0,\ k > 0$$
$$= p \qquad i = 0.\ 0 \leq k \leq a$$
$$= 0 \qquad i = 0,\ k = 0$$

$$Y_{2(i,k)} = p \qquad i = 0,\ k = 0$$
$$= 0 \qquad \text{otherwise}$$

We are interested in Y_1 - the probability that gc occurs at the next action - and Y_2 - the probability that the system will be full at the next action.

$$Y_1 = p \cdot \sum_{i=0}^{a} u_{i0}$$

$$Y_2 = p \cdot u_{00}$$

5.2. Approximation of the stationary distribution

5.2.1. *Stop criteria*

The stationary distribution is a solution of

(5.3) $M \cdot u = 0$

where M is defined in (5.2). As we do not know a method to solve these equations analytically we developed some algorithms to solve them numerically. These algorithms are iterative in nature. A problem of an iterative process is does it converge and when must it be stopped. The following stop criteria turned out to work satisfactorily:

1. Let

$$v_j = \sum_{i+k=j} u_{ik}.$$

v_j is the probability that j units are available. Evidently v_j satisfies equation (4.1). Also

$$v_0 = u_{00}$$

This yields our first criterium:

$$u_{00}^{-1} = \frac{z^{a+2} - z^{a+1-d}}{(1-z)^2} + \frac{d+1}{1-z} \qquad z \neq 1$$

$$= \tfrac{1}{2}(2a+2-d)(d+1) \qquad z = 1$$

2. Define a state

$$[X] = \left\{ \bigcup_{i=0}^{a} [i,0] \right\}$$

The system is in state [X] when it is in one of the states [i,0] ($0 \leq i \leq a$). The probability of being in the state X equals

$$u_X = \sum_{i=0}^{a} u_{i0}.$$

The probability of leaving this state [i.e. the probability that gc or full occurs] equals

$$u_g = p \cdot u_X$$

The mean number of actions between two consecutive occurrences of garbage collection or full t_g equals

$$t_g = \left(\frac{u_{00}}{u_X} \cdot (d+1) + \sum_{i=1}^{a} \frac{u_{i0}}{u_X} \cdot i \right) p^{-1}$$

Clearly

$$u_g \cdot t_g = 1$$

This yields criteria 2:

$$u_{00}(d+1) + \sum_{i=1}^{a} u_{i0} \cdot i = 1.$$

3. Define a state

$$[Y] = \left\{ \bigcup_{i=0}^{a} [0,i] \right\}$$

Let

$$u_Y = \sum_{i=0}^{a} u_{0i}$$

When the system is steady the probability of leaving state x equals the probability of entering the state Y. This implies:

$$r(u_Y - u_{0a}) = p(u_X - u_{00})$$

This yields criteria 3:

$$\frac{u_X - u_{00}}{u_Y - u_{0a}} = z$$

In the next chapters we discuss the algorithms.

5.2.2. *Repeated multiplication with the transition matrix*

We use the fact that the stationary distribution u equals to

$$u = \lim_{u \to \infty} u^{(n)} = \lim_{N \to \infty} P^N u^{(0)}$$

where $u^{(0)}$ is arbitrary, but normalized.

The algorithm is [cf. 8.1.1]:

step 1 store the initialization distribution $u^{(0)}$ into array f.

step 2 <u>while</u> criterium 1 on f does not hold <u>do</u> multiply f by P^2.

The multiplication with P is executed by the procedure *trans*

5.2.3. *A numerical approach*

In program 8.1.2 we solve equation (5.3) by using the following iterative process: From a distribution over the states [0,i] $0 \le i \le a$ we can compute the distribution over all other states [i,k]. Looking at picture (5.1) we compute the distribution column by column from left to right and in each column from top to bottom. From an initial distribution $\{u_{0k}^{(0)}\}_k$ $0 \le k \le a$ we first compute the complete distribution $\{u_{ik}^{(0)}\}_{i,k}$. Then we find our next iterant from the equations:

(5.4) 1 $u_{0a}^{(N)} = pu_{0,0}^{(N-1)}$ $\quad d = a$

$u_{0a}^{(N)} = u_{0a}^{(N-1)}$ $\quad d \neq a$

(5.5) 2 $u_{0i}^{(N)} = pu_{0,i+1}^{(N-1)} + pu_{i+0,1}^{(N-1)} + \Delta_{id}\, pu_{0,0}$

When $u_{i,k}^{(u)} = u_{i,k}^{(u-1)}$ these equations hold.

To eliminate exponential decrease or increase of our iterant we renormalize the new iterant dividing it by

$$\sum_{i,k} u_{ik}$$

After each iteration criterium 1 is applied.

In the program we use array y[0:a] to store the current iterant:

$$y[i] = u_{0,a-i}^{(N)}$$

In the beginning of each iteration the array y is copied into c. Now the columns are computed:

1 Let column i be computed. Then c equals

$c[a-i-k] = u_{ik}$ $\quad 0 \le k \le a-i$

$c[a-k] = u_{k0}$ $\quad 0 \le k \le i$

To compute c we use the statements:

(5.6) c[0] := zc[1]

(5.7) c[k] := (c[k-i] + z * c[k-1]) / (z+1)

(5.6) and (5.7) are derived from (5.1.2) and (5.1.1) respectively. After the columns have been computed this yields:

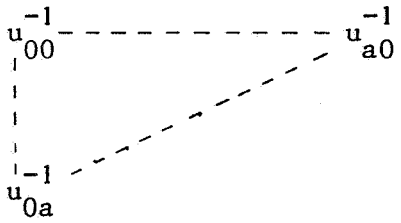
$$c[a-k] = u_{k0} \qquad 0 \leq k \leq a$$

2 Using equations (5.4) and (5.5) we compute the next iteraut.

3 If criteria 1 does not hold the process is repeated.

Note:

The table quoting the mean recurrence time of the states [f,s] are organized as follows:



5.3. Simulation program

Prog. (8.1.3) simulates a system with garbage collection. Each action is either a return or a request of one unit. The state of the simulated system is stored in the variables f and s. F, respectively s, denotes the number of units of the free, respectively semi-free, units. At the beginning of each run they are initialized as:

$$s = 0 \ . \ f = d$$

At each action random is called to determine whether this action will be a request or a return. The number of actions per simulated run is indicated by *it*.

6. DEVIATION OF THE AVERAGE

In the previous chapters we were interested in the average behaviour of some storage allocation systems. In this chapter we pay attention to the de-

viation from the average. We restrict ourselves to the al-system.

We found already that the recurrence time of full equals

$$t_0 = \left(\frac{z^{a+2} - z^{a+1-d}}{(1-z)^2} + \frac{d+1}{1-z}\right)\frac{z+1}{z} \qquad (z \neq 1)$$

$$= (d+1)(2a+2-d) \qquad z = 1$$

where

a	the number of units of the system
d	the number of units made available when the system is full
$z = r.p^{-1}$	r: probability of a return of one block
	p: probability of a request of one block

[i] $0 \leq i \leq a$ represents the state of the system. In state [i] i units are available. We add one auxiliar state [-1] to the system. The actions of the system are numbered such that the system is full at the zeroth action. Initially the system is in state [d]. Let $v^{(N)}$ be the probability that the first time the system is full again at the Nth action (N>0). Clearly:

$$(1) \qquad \tau_0 = \sum_{i=1}^{\infty} i \,.\, v^{(i)}$$

$$(2) \qquad \sum_{i=1}^{\infty} v^{(i)} = 1$$

As we only are interested in the first occurrence of full we discard the behaviour of the system after this first occurrence. This is achieved by entering the system in [-1] when the system is full. Let $(u_i^{(u)})_{i=-1}^{a}$ be the probability that the system is in state i after u actions.

$$(6.1) \qquad \text{initially:} \quad u_i^{(0)} = 0 \qquad i \neq d$$

$$u_d^{(0)} = 1 \qquad i = d$$

The following equations determine the behaviour of the system:

$$
\begin{aligned}
u_i^{(N)} &= pu_{i+1}^{(N-1)} + ru_{i-1}^{(N-1)} \\
u_a^{(N)} &= pu_a^{(N-1)} + ru_{a-1}^{(N-1)} \\
u_0^{(N)} &= pu_1^{(N-1)} \\
u_{-1}^{(N)} &= u_{-1}^{(N-1)} + pu_0^{(N-1)}
\end{aligned}
\qquad (6.2)
$$

Clearly

$$v^{(N)} = p \cdot u_0^{(N)}$$

Let

$$V^{(N)} = \sum_{i=1}^{N} v^i$$

V^u can be calculated from equations 6.2 using the initial distribution 6.1.

The model given above reflects the classical one-dimensional random walk problem with an absorbing and a reflecting state. In the limit $a \to \infty$ an analytic expression for $v^{(u)}$ is known (cf. [4], ch. 3.7., p.132):

$$v^{(2i+d+1)} = \left(\binom{2i+d}{i} - \binom{2i+d}{i-1}\right) 2^{-(2i+d+1)} \qquad i = 1,2,3,\ldots$$

$$= 0 \qquad \text{otherwise}$$

Results:

In the figures of chapter 8.2 $V^{(N)}$ is plotted against N. The average is indicated by a *.

Fig. 8.2.1 shows the curves the highest one corresponds to

a = 16 d = 0 z = 1

The next one corresponds to

a = 32 d = 0 z = 1

The third one corresponds to

a = ∞ d = 0 z = 1

Fig 8.2.2 shows curves of systems with

a = 48 d = 48 z = 0.80, 0.90, 1.00

Fig. 8.2.3 shows curves of systems with

$$a = 48 \qquad z = 1 \qquad d = 0, 12, 24, 36, 48.$$

7. FINAL REMARKS

From the preceding chapters two conclusions must be drawn:

- The behaviour of the systems described heavily depends on the value of z, i.e. the ratio of the probabilities of request and return. Fluctuations of 0.5 percent in z may cause changes of about fifty percent in important system quantities as the mean recurrence time of full and garbage collection. In an actual DSA-system changes of several percent in z are not exceptional. This makes each short time prediction about the occurrences of full and garbage collection impossible.
- We derived the mean recurrence time of full as a function of z. In chapter 5 we showed that the mean recurrence time of full is an average of a very large distribution. The distribution is so large that no practical system will ever execute enough actions to guarantee that the average recurrence time of full is in accordance with the theoretical value. To quote data:
  We simulate a small al-system with 48 units. After 60,000 actions we found deviations of more than 25% from the theoretical value [cf.ch.8.1.3].

8. APPENDIX

8.1. Programs

8.1.1.

[cf. ch. 5.2.2]

```
2506?4- 15      B 2355F.0 CLIPPEL

B 2355F.0,CL P PPEL,R50,T150

        'BEGIN'
    1     'COMMENT'
    2       THIS PROGRAM COMPUTES THE STATIONARY DISTRIBUTION OF A DSA-SYSTEM
    3       WITH GARBAGE COLLECTION BY  REPEATED MULTIPLICATION OF THE STOCHASTIC MATRIX;
    4     'INT' A,MAXIT,D;
    5     'REAL' P,Q,R,Z,EPS;
    6     'PROC' IN(S,N);'STRING' S;'REAL' N;
    7     'BEGIN'
    8       N:=READ;
    9       NLCR;PRINTTEXT(S);TAB;ABSFIXT(4,3,N);
   10     'END';
   11     'PROC' OUT(S,N);'STRING' S;'REAL' N;
   12     'BEGIN'
   13       NLCR;PRINTTEXT(S);TAB;ABSFIXT(4,3,N);
   14     'END';
   15     'EXT':
   16       IN("A",A);IN("D",D);IN("Z",Z);IN("Q",Q);IN("MAXIT",MAXIT);IN("EPS",EPS);
   17       P:=(1-Q)/(1+Z);
   18       R:=Z*P;
   19       'BEGIN'
   20         'REAL''ARRAY' F,G[0:A,0:A];
   21         'BOOL' FLIP;
   22         'INT' I,K,PT,NEP,IT;
   23         'REAL' S,SUM,FE,DIF,GT,SX;
   24         'PROC' TRANS;
   25         'BEGIN'
   26           FLIP:=¬FLIP;
   27           'IF' FLIP
   28             'THEN'
   29             'BEGIN'
   30               'FOR' K:=0 'STEP' 1 'UNTIL' A-1 'DO'
   31                 F[0,K]:=P*G[0,K+1]+P*G[K+1,0]+Q*G[0,K];
   32               F[0,A]:=(1-P)*G[0,A];
   33               F[0,0]:=F[0,0]+P*G[0,0];
   34               'FOR' I:=1 'STEP' 1 'UNTIL' A 'DO'
   35                 'BEGIN'
   36                   'FOR' K:=0 'STEP' 1 'UNTIL' A-I-1 'DO'
   37                     F[I,K]:=R*G[I-1,K]+P*G[I,K+1]+Q*G[I,K];
   38                   F[I,A-I]:=(1-P)*G[I,A-I]+R*G[I-1,A-I];
   39                 'END'
   40             'END'
   41             'ELSE'
   42             'BEGIN'
   43               'FOR' K:=0 'STEP' 1 'UNTIL'A-1 'DO'
   44                 G[0,K]:=P*F[0,K+1]+P*F[K+1,0]+Q*F[0,K];
   45               G[0,A]:=(1-P)*F[0,A];
   46               G[0,0]:=G[0,0]+P*F[0,0];
   47               'FOR' I:=1 'STEP' 1 'UNTIL' A 'DO'
   48                 'BEGIN'
   49                   'FOR' K:=0 'STEP' 1 'UNTIL' A-I-1 'DO'
   50                     G[I,K]:=R*F[I-1,K]+P*F[I,K+1]+Q*F[I,K];
   51                   G[I,A-I]:=(1-P)*F[I,A-I]+R*F[I-1,A-I];
   52                 'END'
   53             'END'
   54         'END';
   55         PT:=0.5*(A+1)*(A+2);
```

```
250674- 15     B 2355F.0 CLPIPPEL

   56       S:='IF' Z=1 'THEN' 0,5*(D+1)*(2*A-D+2)
   57                   'ELSE' (Z**(A+2)-Z**(A+1-D))/(Z-1)**2 - (D+1)/(Z-1);
   58       FE:=1/S;
   59       'FOR' I:=0 'STEP' 1 'UNTIL' A 'DO'
   60         'FOR' K:=0 'STEP' 1 'UNTIL' A-I 'DO' F[I,K]:=1/PT;
   61       IT:=0;DIF:=1;FLIP:='TRUE';
   62                                  'FOR' NEP:=0
   63       'WHILE' ABS(DIF)>EPS ^ IT<MAXIT 'DO'
   64         'BEGIN'
   65           IT:=IT+1;
   66           TRANS;TRANS;
   67           DIF:=(F[0,0]-FE)/FE;
   68         'END';
   69       NEW PAGE;
   70       PRINTTEXT("RECURRENCE TIME OF [S,F]");
   71       NLCR;
   72       'FOR' I:=0 'STEP' 1 'UNTIL' A 'DO'
   73         'BEGIN'
   74           NLCR;
   75           'FOR' K:=0 'STEP' 1 'UNTIL' A-I 'DO'
   76             'BEGIN'
   77               'IF' F[I,K]>0.0001
   78                 'THEN' ABSFIXT(4,1,1/F[I,K])
   79                 'ELSE' PRINTTEXT(" ****** ");
   80             'END'
   81         'END';
   82       SX:=0;
   83       'FOR' I:=0 'STEP' 1 'UNTIL' A 'DO'
   84         SX:=SX+F[I,0];
   85       GT:=0;
   86       GT:=F[0,0]*(D+1);
   87       'FOR' I:=1 'STEP' 1 'UNTIL' A 'DO'
   88         GT:=GT+F[I,0]*I;
   89       GT:=GT/(SX*P);
   90       NEW PAGE;
   91       OUT("ITERATIONS",2*IT);
   92       OUT("REL ERROR",DIF);
   93       OUT("MEAN RT [0,0]",1/F[0,0]);
   94       OUT("MEAN RT [0,D]",1/F[0,D]);
   95       OUT("MEAN RT GC",GT);
   96       OUT("PROBABILITY GC",SX*P);
   97       OUT("PRODUCT",GT*SX*P);
   98     'END';
   99     'IF' READ>0 'THEN'
  100       'BEGIN'
  101         NEW PAGE;
  102         'GOTO' NEXT;
  103       'END';
  104   'END';
```

250674- 15 B 2355F.0 CLIPPEL

A	10.000
D	.000
Z	1.000
Q	.000
MAXIT	150.000
EPS	.005

250674- 15 B 2355F.0 CLPIPPEL

RECURRENCE TIME OF (S,F)

17.1	48.1	63.9	80.7	98.0	115.5	132.8	149.9	166.6	182.8	198.3	213.4	229.8	252.7	298.2	447.3	******
28.5	58.9	75.9	93.6	111.5	129.2	146.7	163.9	180.7	197.4	214.6	233.9	258.8	296.2	357.8	447.2	
38.6	70.9	86.9	107.2	125.4	143.4	161.1	178.5	196.1	214.3	234.3	258.0	287.7	324.0	357.7		
52.8	83.9	102.7	121.4	139.9	158.1	176.1	194.2	212.8	232.8	254.9	279.4	304.8	324.0			
66.6	97.7	117.0	136.1	154.8	173.3	191.8	210.7	230.3	250.8	271.9	291.5	304.7				
85.9	112.3	132.0	151.3	170.2	189.1	208.0	227.2	246.5	265.2	281.3	291.4					
102.1	127.5	147.4	166.8	186.0	204.9	223.7	242.0	259.0	273.0	281.3						
119.2	143.1	163.1	182.6	201.6	220.0	237.6	253.3	265.8	272.9							
136.2	159.0	178.8	198.0	216.2	233.2	248.0	259.4	265.7								
155.1	174.8	194.1	212.2	228.8	242.9	253.5	259.3									
169.6	190.0	208.1	224.4	238.0	248.0	253.5										
185.1	203.9	219.9	233.2	242.9	248.0											
199.2	215.5	228.6	237.9	242.8												
210.8	223.9	233.1	237.9													
219.2	228.4	233.1														
223.7	228.4															
223.7																

250674- 15 B 2355F.0 CLIPPEL

ITERATIONS	294.000
REL ERROR	.005
MEAN RT [0,0]	17.083
MEAN RT [0,0]	17.083
MEAN RT GC	6.541
PROBABILITY GC	.117
PRODUCT	1.000

8.1.2.

[cf. ch. 5.2.3]

```
21...74+ 35     A 2355F.0 CLP PPEL

A2355F.0,CL P PPEL

      'BEGIN'
        'COMMENT'
          THIS PROGRAM COMPUTES THE STATIONARY DISTRIBUTION OF A DSA-SYSTEM
          WITH GARBAGE COLLECTION,USING A METHOD OF SUCCESSIVE ITERATIONS;
        'REAL' Z,P,Q,R,EPS;
        'INT' A, T,D;
        'PROC' IN(S,N); 'STRING'S;'REAL'N;
        'BEGIN'
          N:=READ;
          NLCR;PRINTTEXT(S);TAB;ABSFIXT(4,3,N);
        'END';
        'PROC' OUT(S,N);'STRING'S;'REAL'N;
        'BEGIN'
          NLCR;PRINTTEXT(S);TAB;FIXT(4,3,N);
        'END';
      NEXT:
        IN("A",A);IN("Z",Z);IN("Q",Q);IN("IT",IT);IN("EPS",EPS);IN("D",D);
        P:=(1-Q)/(Z+1);R:=Z*P;
        'BEGIN'
          'INT' ,K,STEP;
          'REAL' S,ER,SUM,STH,SX,FX,GR,UOD,SY,QU;
          'REAL''ARRAY' C,Y[0:A];
          'PROC' NEW(COL,N);'VAL' N;'REAL''ARRAY' COL;'INT'N;
          'BEGIN'
            'INT' K;
            COL[0]:=Z*COL[1];
            'FOR' K:=1 'STEP' 1 'UNTIL' N 'DO'
              COL[K]:=(COL[K-1]+Z*COL[K+1])/(Z+1);
          'END';
          'REAL''PROC' DIAG(N);'INT'N;
          'BEGIN'
            'REAL''PROC' DG(P);'INT'P;
            DG:='IF' Z=1
                  'THEN' P+1
                  'ELSE' (Z**(P+1)-1)/(Z-1);
            DIAG:=('IF'N 'LE' D
                     'THEN' DG(N)
                     'ELSE' Z**(N-D)*DG(D)
                  )/S;
          'END';
          S:='IF' Z'NE'1
                'THEN'(Z**(A+2) - Z**(A+1-D))/(Z-1)**2 - (D+1)/(Z-1)
                'ELSE'0.5*(D+1)*(2*A-D+2);
          'FOR' I:=0 'STEP' 1 'UNTIL' A 'DO'
            Y[A-I]:=DIAG(I)/(I+1);
          ER:=1;
          'FOR'K:=1,K+1 'WHILE' K<IT^ ABS(ER)>EPS 'DO'
            'BEGIN'
              'INT' I;
              STEP:=K;
              'FOR' I:=0 'STEP' 1 'UNTIL' A 'DO' C[I]:=Y[I];
              SUM:=0;
              'FOR' I:=A-1 'STEP' -1 'UNTIL' 0 'DO'
                'BEGIN'
                  'INT'K;
                  'FOR' K:=0 'STEP' 1 'UNTIL' I+1 'DO' SUM:=SUM+C[K];
```

```
2106 4- 35     A 2355F.0 CLPIPPEL

                NEW(C,I);
              'END';
            SUM:=SUM+C[0];
            STH:=Y[A]*S;
            ER:=(STH-SUM)/STH;
            Y[0]:=('IF' D=A 'THEN' 1 'ELSE' 0)*Y[A];
            'FOR' I:=1 'STEP' 1 'UNTIL' A 'DO'
              Y[I]:=(Y[I-1]+C[I-1]+Y[A]*('IF' D=A-I 'THEN'1 'ELSE'0))/(Z+1);
          'END';
          'FOR' I:=0 'STEP' 1 'UNTIL' A 'DO'
            Y[ ]:=Y[I]/SUM;
          U0D:=Y[A-D];
          NEW PAGE;
          PRINTTEXT("MEAN RECURRENCE TIME OF [S,F]");
          NLCR;
          'FOR' I:=A'STEP'-1 'UNTIL' 0 'DO'
            'BEGIN'
              C[I]:=Y[I];
              'IF' C[I]'NE'0
                'THEN' ABSFIXT(4,1,1/C[I])
                'ELSE' PRINTTEXT(" ****** ");
            'END';
          'FOR' I:=A-1'STEP'-1'UNTIL'0'DO'
            'BEGIN'
              'INT'K;
              NEW(C,I);
              NLCR; NLCR;
              'FOR' K:=I 'STEP' -1 'UNTIL' 0 'DO'
                'IF' C[K]'NE'0
                  'THEN' ABSFIXT(4,1,1/C[K])
                  'ELSE' PRINTTEXT(" ****** ");
            'END';
          SX:=0;
          'FOR' I:=0 'STEP' 1 'UNTIL' A 'DO' SX:=SX+C[I];
          SY:=0;
          'FOR' I:=0 'STEP' 1 'UNTIL' A 'DO' SY:=SY+Y[I];
          QU:=(SX-C[A])/(SY-Y[0]);
          GR:=C[A]*(D+1);
          'FOR' I:=1 'STEP' 1 'UNTIL' A 'DO' GR:=GR+C[A-I]*I;
          GR:=GR/(P*SX);
          NEW PAGE;
          OUT("ITERATIONS",STEP);
          OUT("REL ERROR",ER);
          OUT("REC TIME [0,0]",1/C[A]);
          OUT("REC TIME [0,D]",1/U0D);
          OUT("REC TIME GC",GR);
          OUT("PROBABILITY GC",SX*P);
          OUT("PRODUCT",GR*SX*P);
          OUT("X/Y      ",QU);
      'END';
      'IF' READ>0 'THEN'
        'BEGIN'
          NEW PAGE;
          'GOTO' NEXT;
        'END';
    'END'
```

210674- 35 A 2355F.0 CLPIPPEL

A	16.000
Z	1.000
Q	.000
[illegible]	100.000
EPS	.005
D	.300

2[illegible]4- 35 A 2355F.0 CLPIPPEL

MEAN RECURRENCE TIME OF [S,F]

17.1	48.0	63.8	80.6	97.9	115.3	132.7	149.8	166.7	182.9	198.5	213.7	230.3	253.2	298.9	448.4	******
26.5	58.8	75.8	93.5	111.4	129.1	146.6	163.9	180.8	197.6	214.9	234.3	259.4	296.9	358.7	448.4	
36.5	70.8	88.8	107.1	125.3	143.3	161.1	178.6	196.3	214.6	234.7	258.6	288.5	324.9	358.7		
52.8	83.7	102.5	121.3	139.8	158.1	176.2	194.4	213.2	233.2	255.4	280.1	305.6	324.9			
68.5	97.6	116.9	136.0	154.8	173.4	192.0	211.0	230.7	251.4	272.6	292.3	305.6				
83.1	112.2	131.9	151.2	170.3	189.2	208.3	227.6	247.1	265.9	282.1	292.3					
102.0	127.4	147.3	166.9	186.1	205.2	224.1	242.6	259.7	273.8	282.1						
119.1	143.0	163.1	182.7	201.8	220.5	238.1	254.0	266.6	273.8							
136.2	159.6	179.0	198.2	216.6	233.7	248.7	260.1	266.6								
153.1	174.9	194.3	212.6	229.3	243.6	254.3	260.1									
169.7	190.2	208.5	224.9	238.6	248.8	254.3										
185.4	204.2	220.5	233.9	243.6	248.8											
199.5	216.0	229.2	238.6	243.6												
211.3	224.6	233.8	238.6													
219.8	229.1	233.8														
224.4	229.1															
224.4																

2136/4- 35 A 2355F.0 CLPIPPEL

ITERATIONS	+16.000
REL ERROR	-.005
REQ TIME (0,0)	+17.072
REQ TIME (0,D)	+17.072
REQ TIME GC	+8.533
PROBABILITY GC	+.117
PRODUCT	+.999
X/Y	+1.000

210674- 35 A 2355F.0 CLIPPEL

A	48.000
Z	1.000
Q	.000
IT	100.000
EPS	.005
D	16.000

21.6[illegible]4- 35 A 2355F.0 CLPIPPEL

MEAN RECURRENCE TIME OF (S,F)

700.1	700.0	699.9	699.7	699.5	699.4	699.4	699.8	700.9	702.6	704.9	706.6	705.2	695.2	667.2	608.2	508.9	824.9
863.1	883.9	917.0	951.9	988.5	1026.4	1065.3	1104.9	1145.2	1185.8	1226.7	1267.7	1308.7	1349.6	1390.2	1430.4	1469.9	1508.5
1548.8	1581.7	1615.7	1647.9	1678.6	1709.1	1742.6	1786.6	1857.8	1998.2	2335.1	3502.7	******					
700.0	699.9	699.8	699.7	699.6	699.7	700.0	700.5	701.2	701.6	700.6	696.4	686.5	668.7	644.1	622.5	637.6	853.2
838.5	910.1	950.7	986.9	1024.5	1063.2	1102.7	1142.8	1183.3	1224.2	1265.1	1306.1	1346.9	1387.5	1427.6	1467.1	1505.6	1543.2
1579.5	1614.6	1648.9	1683.6	1720.9	1765.5	1825.7	1917.2	2068.4	2332.8	2802.2	3502.7						
699.9	699.8	699.7	699.7	699.7	699.8	699.9	699.9	699.3	697.3	693.0	685.6	675.2	664.3	659.9	676.4	740.5	813.1
915.3	949.4	985.3	1022.6	1061.1	1100.4	1140.5	1180.9	1221.7	1262.6	1303.5	1344.3	1384.8	1424.9	1464.4	1503.1	1541.0	1578.1
1614.6	1651.2	1689.6	1732.3	1783.8	1851.5	1946.3	2083.9	2282.5	2546.1	2802.2							
699.8	699.7	699.7	699.6	699.5	699.3	698.9	697.8	695.8	692.3	687.3	681.7	677.9	680.7	697.9	740.7	818.4	914.6
948.3	983.9	1 20.9	1059.1	1098.3	1138.1	1178.5	1219.2	1260.1	1301.0	1341.7	1382.2	1422.3	1461.9	1500.9	1539.3	1577.2	1615.1
1653.9	1695.3	1 41.8	1797.3	1867.4	1959.2	2080.2	2233.9	2407.1	2546.1								
699.7	699.6	699.4	699.2	698.9	698.2	697.1	695.3	692.9	690.0	687.7	688.2	694.7	712.3	747.0	803.6	878.2	947.3
982.5	1019.2	1057.2	1096.2	1135.9	1176.1	1216.8	1257.6	1298.4	1339.2	1379.7	1419.9	1459.6	1498.9	1537.8	1576.7	1615.9	1656.6
1700.2	1749.2	1806.7	1876.6	1963.1	2069.2	2192.3	2317.3	2407.1									
699.5	699.3	699.0	698.6	697.9	696.9	695.6	694.2	693.0	693.2	696.3	705.1	722.9	753.6	799.9	860.8	926.6	981.2
1015.6	1055.3	1094.1	1133.6	1173.8	1214.3	1255.1	1295.9	1336.7	1377.2	1417.5	1457.5	1497.1	1536.6	1576.3	1616.7	1658.9	1704.2
1754.7	1812.8	1881.1	1962.0	2055.6	2157.2	2253.1	2317.3										
699.2	698.9	698.4	697.9	697.1	696.4	695.9	696.1	698.0	703.1	713.4	731.3	759.5	799.9	852.2	911.9	969.4	1016.1
1053.5	1092.1	1131.5	1171.5	1211.9	1252.6	1293.4	1334.2	1374.8	1415.3	1455.5	1495.5	1535.5	1576.0	1617.4	1660.8	1707.3	1758.5
1816.4	1882.6	1957.9	2041.3	2127.3	2204.1	2253.1											
698.9	698.5	698.0	697.7	697.5	697.8	699.1	702.5	708.9	720.2	738.1	764.7	801.2	847.9	902.0	957.9	1008.9	1051.8
1[illegible]	1129.3	1169.[illegible]	1209.6	1250.2	1291.0	1331.8	1372.5	1413.1	1453.5	1493.9	1534.5	1575.7	1617.9	1662.1	1709.4	1760.9	1818.1
1881.8	1954.2	2027.1	2101.3	2165.0	2204.1												
698.6	698.4	698.4	698.7	699.8	702.1	706.5	714.0	726.0	743.9	769.3	803.2	845.7	895.5	948.7	1000.5	1047.1	1088.2
1[illegible]	1161.[illegible]	1207.3	1247.8	1288.6	1329.5	1370.3	1411.0	1451.7	1492.5	1533.5	1575.2	1618.2	1663.0	1710.7	1762.2	1816.3	1879.6
1945.4	2013.4	2078.6	2132.7	2165.0													
699.9	699.3	700.1	701.8	705.0	710.3	718.6	731.1	748.9	773.4	805.3	844.8	891.1	941.6	992.7	1040.9	1084.7	1125.2
1[illegible].8	1205.3	1245.5	1286.3	1327.1	1368.0	1408.9	1449.9	1491.0	1532.5	1574.7	1618.1	1663.4	1711.3	1762.4	1817.4	1876.3	1938.0
2[illegible].3	2058.3	2105.3	2132.7														
7[illegible].3	701.6	703.9	707.7	713.7	722.7	735.6	753.4	777.1	807.5	844.7	888.1	936.1	985.9	1034.6	1080.1	1122.4	1162.7
1[illegible].7	1243.2	1284.0	1324.9	1365.9	1406.9	1448.1	1489.5	1531.4	1574.0	1617.8	1663.4	1711.3	1761.9	1815.7	1872.2	1930.4	1987.6
2[illegible].1	2081.5	2105.3															
7[illegible].1	705.9	710.4	717.0	726.4	739.6	757.4	780.5	809.6	845.0	886.1	931.9	980.1	1028.5	1075.0	1118.7	1160.1	1200.5
1[illegible].0	1281.7	1322.7	1363.7	1405.0	1446.3	1488.0	1530.2	1573.1	1617.3	1663.0	1710.7	1760.8	1813.2	1867.6	1922.6	1975.9	2023.5
2[illegible].6	2081.5																
[illegible].[illegible]	712.9	720.0	729.9	743.3	761.0	783.6	811.7	845.5	884.8	928.6	975.3	1023.0	1069.8	1114.5	1157.1	1198.2	1238.8
1[illegible].5	1320.5	1361.6	1403.0	1444.6	1486.5	1528.9	1572.1	1616.4	1662.2	1709.7	1759.1	1810.3	1862.7	1914.9	1964.6	2008.3	2041.9
2[illegible].6																	
7[illegible].3	722.9	733.1	746.7	764.3	786.5	813.7	846.3	884.0	926.0	971.3	1018.1	1064.8	1110.1	1153.6	1195.4	1236.5	1277.3
1[illegible].3	1359.6	1401.0	1442.8	1484.9	1527.6	1571.0	1615.4	1661.1	1708.3	1757.0	1806.9	1857.5	1907.1	1953.8	1994.3	2025.0	2041.9
7[illegible].6	736.1	749.8	767.3	789.2	815.7	847.2	883.4	924.0	967.9	1013.7	1060.0	1105.6	1149.8	1192.4	1234.0	1275.1	1316.2
1[illegible].5	1399.1	1441.0	1483.3	1526.1	1569.6	1614.1	1659.7	1706.5	1754.5	1803.3	1852.1	1899.5	1943.5	1981.3	2009.5	2025.0	
7[illegible].9	752.8	770.2	791.7	817.6	848.1	883.2	922.4	965.0	1009.9	1055.7	1101.3	1145.9	1189.1	1231.3	1272.7	1314.0	1355.4

210674- 35 A 2355F.0 CLPIPPEL

[illegible]	1439.2	1481.6	1524.5	1568.2	1612.6	1658.0	1704.4	1751.7	1799.3	1846.6	1892.0	1933.7	1969.1	1995.3	2009.5			
[illegible]	[illegible]	794.0	819.4	849.1	883.1	921.2	962.6	1006.5	1051.7	1097.1	1141.9	1185.7	1228.4	1270.3	1311.8	1353.4	1395.2	
[illegible]	1479.8	1522.9	1566.6	1611.0	1656.2	1702.1	1748.7	1795.2	1841.0	1884.6	1924.2	1957.6	1982.1	1995.3				
[illegible]	796.2	821.1	850.1	883.2	920.3	960.6	1003.5	1048.0	1093.2	1138.0	1182.1	1225.3	1267.7	1309.5	1351.3	1393.2	1435.4	
[illegible]	1541.2	1584.8	1609.2	1654.1	1699.6	1745.4	1791.0	1835.4	1877.3	1915.1	1946.7	1969.8	1982.1					
[illegible]	822.8	851.1	883.4	919.5	958.9	1000.9	1044.7	1089.5	1134.3	1178.6	1222.1	1264.9	1307.1	1349.1	1391.2	1433.5	1476.2	
[illegible]	1565.0	1607.2	1651.9	1696.9	1742.0	1786.6	1829.7	1870.2	1906.4	1936.4	1958.2	1969.8						
[illegible]	852.2	883.7	918.9	957.4	998.6	1041.7	1086.0	1130.6	1175.0	1218.9	1262.0	1304.6	1346.9	1389.1	1431.5	1474.3	1517.4	
[illegible]	1589.1	1649.5	1694.1	1738.5	1782.1	1824.1	1863.2	1898.0	1926.6	1947.2	1958.2							
[illegible]	884.1	918.5	956.2	996.5	1039.0	1082.8	1127.2	1171.6	1215.6	1259.0	1301.9	1344.5	1386.9	1429.5	1472.3	1515.4	1559.0	
[illegible]	1690.9	1691.1	1734.8	1777.6	1818.5	1856.4	1889.8	1917.2	1936.9	1947.2								
[illegible]	918.2	955.1	994.7	1036.5	1079.7	1123.9	1168.2	1212.3	1256.0	1299.2	1342.0	1384.7	1427.4	1470.2	1513.4	1556.8	1600.5	
[illegible]	1688.0	1731.1	1773.0	1812.9	1849.7	1882.0	1908.2	1927.0	1936.9									
[illegible]	954.2	993.1	1034.2	1076.9	1120.7	1164.9	1209.1	1253.0	1296.4	1339.5	1382.4	1425.2	1468.1	1511.2	1554.5	1598.0	1641.5	
[illegible]	1727.2	1768.4	1807.4	1843.1	1874.3	1899.6	1917.6	1927.0										
[illegible]	991.6	1032.1	1074.3	1117.7	1161.8	1205.9	1249.9	1293.6	1336.9	1379.9	1422.9	1465.9	1509.0	1552.2	1595.5	1638.7	1681.5	
[illegible]	1763.8	1801.9	1836.6	1866.9	1891.3	1908.5	1917.6											
[illegible]	1030.2	1071.9	1114.9	1158.7	1202.8	1246.9	1290.7	1334.2	1377.4	1420.5	1463.6	1506.7	1549.8	1592.9	1635.8	1678.1	1719.4	
[illegible]	1796.4	1830.3	1859.6	1883.2	1899.9	1908.5												
[illegible]	1069.7	1112.3	1155.8	1199.8	1243.8	1287.8	1331.4	1374.9	1418.1	1461.2	1504.3	1547.3	1590.2	1632.7	1674.6	1715.4	1754.5	
[illegible]	1924.1	1862.6	1875.4	1891.5	1899.9													
[illegible]	1109.8	1153.0	1196.8	1240.8	1284.8	1328.6	1372.2	1415.6	1458.7	1501.8	1544.7	1587.4	1629.7	1671.2	1711.4	1749.8	1785.7	
[illegible]	1845.7	1867.9	1883.4	1891.5														
[illegible]	1150.3	1193.9	1237.9	1281.9	1325.8	1369.5	1413.0	1456.2	1499.3	1542.1	1584.6	1626.5	1667.6	1707.4	1745.2	1780.4	1811.9	
[illegible]	1860.6	1875.6	1883.4															
[illegible]	1191.2	1235.0	1279.0	1323.0	1366.8	1410.3	1453.6	1496.6	1539.4	1581.7	1623.3	1664.0	1703.3	1740.6	1775.1	1806.0	1832.5	
[illegible]	1868.1	1875.6																
[illegible]	1232.2	1276.2	1320.1	1364.0	1407.6	1450.9	1493.9	1536.6	1578.7	1620.1	1660.4	1699.2	1736.0	1769.9	1800.2	1826.0	1846.5	
[illegible]	1868.1																	
[illegible]	[illegible]	1317.3	1361.2	1404.8	1448.2	1491.2	1533.7	1575.7	1616.8	1656.8	1695.2	1731.4	1764.7	1794.5	1819.7	1839.7	1853.6	
[illegible]																		
[illegible]	1314.4	1358.3	1402.0	1445.4	1488.3	1530.8	1572.6	1613.4	1653.1	1691.1	1726.8	1759.6	1788.8	1813.6	1833.1	1846.6	1853.6	
[illegible]	1355.5	1399.1	1442.5	1485.4	1527.8	1569.4	1610.1	1649.4	1686.9	1722.2	1754.6	1783.2	1807.5	1826.6	1839.8	1846.6		
[illegible]	1396.3	1439.6	1482.5	1524.8	1566.2	1606.6	1645.6	1682.8	1717.7	1749.5	1777.7	1801.6	1820.3	1833.2	1839.8			
[illegible]	1436.7	1479.5	1521.7	1563.0	1603.2	1641.9	1678.7	1713.1	1744.6	1772.3	1795.7	1814.0	1826.7	1833.2				
[illegible]	[illegible]	1518.5	1559.7	1599.7	1638.1	1674.6	1708.6	1739.6	1767.0	1790.0	1807.9	1820.3	1826.7					
[illegible]	1515.4	1556.4	1596.1	1634.3	1670.5	1704.1	1734.8	1761.7	1784.3	1802.0	1814.1	1820.3						
[illegible]	1553.0	1592.6	1630.5	1666.3	1699.7	1729.9	1756.5	1778.7	1796.1	1808.0	1814.1							

210674- 35 A 2355F.0 CLPIPPEL

1549.6	1589.0	1626.6	1662.2	1695.2	1725.1	1751.3	1773.2	1790.3	1802.0	1808.0
1565.4	1622.8	1658.1	1690.8	1720.3	1746.2	1767.8	1784.6	1796.1	1802.0	
1618.9	1654.0	1686.3	1715.6	1741.1	1762.4	1779.0	1790.4	1796.1		
1639.8	1681.9	1710.9	1736.1	1757.2	1773.5	1784.7	1790.4			
1677.5	1706.2	1731.2	1751.9	1768.0	1779.1	1784.7				
1701.5	1726.2	1746.8	1762.7	1773.5	1779.1					
1721.4	1741.7	1757.4	1768.1	1773.5						
1736.6	1752.1	1762.7	1768.1							
1746.9	1757.4	1762.7								
1752.2	1757.4									
1752.2										

210674- 35 A 2355F.0 CLIPPEL

ITERATIONS	+20.000
REL ERROR	-.005
REC TIME [0,0]	+700.135
REC TIME [0,D]	+508.933
REC TIME GC	+39.863
PROBABILITY GC	+.025
PRODUCT	+1.000
X/Y	+1.000

8.1.3.

[cf. ch. 5.3]

```
21[illegible]4- 22     B 2355F.0 CLPIPPEL

B2[illegible]5F.0,CL PIPPEL,T150

      'BEGIN'
        'COMMENT'
          TH S PROGRAM S MULATES A DSA-SYSTEM WITH GARBAGE COLLECTION;
        'REAL' Z,P,GC,FULL;
        'INT' NEP;
        'INT' A,D,GCCTR,FCTR,ITCTR,F,S,IT,R CTR,MR;
        'PROC' IN(S,N); 'STRING'S;'REAL'N;
        'BEGIN'
          N:=READ;
          NLCR;PRINTTEXT(S);TAB;ABSFIXT(6,3,N);
        'END';
        'PROC' OUT(S,N);'STRING'S;'REAL'N;
        'BEGIN'
          NLCR;PRINTTEXT(S);TAB;FIXT(6,3,N);
        'END';
      NEXT:
        R CTR:=0;
        SET RANDOM(0.5);
         IN("A",A);IN("Z",Z);IN("D",D);IN("IT",IT);IN("MR",MR);
      NEXT2:
        R CTR:=R CTR+1;
        NLCR;
        OUT("RUN:",R CTR);
        P:=1/(Z+1);
        S:=0;F:=D;
        ITCTR:=0;FCTR:=0;GC CTR:=0;GC:=0;FULL:=0;
                                              'FOR' NEP:=0
        'WHILE' ITCTR<IT 'DO'
          'BEGIN'
            ITCTR:=ITCTR+1;
            'IF' RANDOM<P
              'THEN'
                'BEGIN'
                  'IF' F<1
                    'THEN'
                      'BEGIN'
                        GCCTR:=GCCTR+1;
                        GC:=IT CTR;
                        F:=F+S-1;
                        S:=0;
                        'IF' F<0
                          'THEN'
                            'BEGIN'
                              F:=F+D;
                              FCTR:=FCTR+1;
                              FULL:=IT CTR;
                            'END';
                      'END'
                    'ELSE' F:=F-1;
                'END'
              'ELSE'
                'BEGIN'
                  'IF' F+S+1'LE'A
                    'THEN' S:=S+1;
                'END';
          'END';
```

```
210674- 22      B 2355F.0 CLPIPPEL

   56      OUT("MEAN REC TIME FULL",FULL/F CTR);
   57      OUT("MEAN REC TIME GC",GC/GC CTR);
   58      'IF' R CTR<MR 'THEN' 'GOTO' NEXT2;
   59      'IF' READ >0 'THEN'
   60        'BEGIN'
   61          NEW PAGE;
   62          'GOTO' NEXT;
   63        'END';
   64    'END';
```

210674- 22 B 2355F.0 CLPIPPEL

```
A              48.000
Z               1.000
D              16.000
IT          60000.000
MR             10.000

RUN:           +1.000
MEAN REC TIME FULL          +1239.128
MEAN REC TIME GC              +39.255

RUN:           +2.000
MEAN REC TIME FULL          +1094.132
MEAN REC TIME GC              +38.507

RUN:           +3.000
MEAN REC TIME FULL          +1542.946
MEAN REC TIME GC              +40.039

RUN:           +4.000
MEAN REC TIME FULL          +1524.846
MEAN REC TIME GC              +40.298

RUN:           +5.000
MEAN REC TIME FULL          +1175.294
MEAN REC TIME GC              +39.259

RUN:           +6.000
MEAN REC TIME FULL          +1110.944
MEAN REC TIME GC              +38.456

RUN:           +7.000
MEAN REC TIME FULL          +1489.487
MEAN REC TIME GC              +38.640

RUN:           +8.000
MEAN REC TIME FULL          +1325.000
MEAN REC TIME GC              +38.501

RUN:           +9.000
MEAN REC TIME FULL          +1753.194
MEAN REC TIME GC              +47.954

RUN:          +10.000
MEAN REC TIME FULL          +1450.341
MEAN REC TIME GC              +40.577
```

```
                                * SOURCE LISTING *          A68  1.0.3  74330  12/12/74  13.22.14.

0001.          !BEGIN!                                                                          1
0002.            #COMPUTE THE RECURRENCE TIME OF FULL#                                          1
0003.            !STRING! TAB="  ";                                                            1
0004.            !PROC! FULL=(!INT! A,D,!REAL! Z) !REAL!;                                      1
0005.            !IF! !ABS!(Z-1)<0.001                                                         2 DENOTATION
0006.              !THEN! (D+1)*(2*A+2-D)                                                      3
0007.              !ELSE! ((Z**(A+2)- Z**(A+1-D))/(1-Z)**2 + (D+1)/(1-Z))*(Z+1)/Z              3       TAG
0008.            !FI!;                                                                         1
0009.            !FOR! A !FROM! 48 !TO! 48                                                     1 DENOTATION
0010.              !DO!                                                                        2
0011.                !INT! DELTA=A!OVER!12;                                                    2
0012.                !REAL! Z:=0.90;                                                           2
0013.                PRINT((NEW PAGE,"RECURRENCE TIME FULL",NEW LINE,"A=",WHOLE(A,0)));       2
0014.                PRINT((NEW LINE,"D    "));                                               2
0015.                !FOR!D !FROM! 0 !BY! DELTA !TO! A                                         2       TAG
0016.                  !DO! PRINT((TAB,WHOLE(D,-8))) !OD!;                                     2
0017.                PRINT((NEWLINE,"Z   /"));                                                 2
0018.                !WHILE! Z !LE!1.101                                                       3 DENOTATION
0019.                  !DO!                                                                    3
0020.                    PRINT((NEW LINE,FIXED(Z,-4,2),"/"));                                  3
0021.                    !FOR! D !FROM! 0 !BY! DELTA !TO! A                                    3       TAG
0022.                      !DO! PRINT((TAB,FIXED(FULL(A,D,Z),-8,1))) !OD!;                     3
0023.                    Z+:=0.01                                                              3 DENOTATION
0024.                  !OD!                                                                    2
0025.              !OD!                                                                        1
0026.          !END!                                                                           0

PROGRAM LENGTH      000620B WORDS
```

RECURRENCE TIME FULL

A=48

D / Z	0	4	8	12	16	20	24	28	32	36	40	44	48
0,90/	21,0	104,8	188,3	271,3	353,5	434,5	513,7	590,2	662,5	728,5	784,9	826,4	845,5
0,91/	23,1	115,2	206,8	297,6	387,2	475,2	560,8	642,9	719,8	789,2	847,6	890,1	909,3
0,92/	25,6	127,8	229,1	329,3	427,7	523,8	616,7	705,0	786,9	860,0	920,6	964,0	983,3
0,93/	28,8	143,3	256,5	367,8	476,6	582,2	683,4	778,7	866,3	943,3	1006,4	1050,7	1070,1
0,94/	32,7	162,6	290,2	415,1	536,3	653,0	763,9	867,2	960,9	1042,2	1107,8	1153,1	1172,6
0,95/	37,7	186,8	332,4	473,8	610,0	739,8	861,7	974,1	1074,6	1160,6	1228,9	1275,2	1294,8
0,96/	44,1	217,7	385,8	547,5	701,7	847,0	981,9	1104,5	1212,6	1303,7	1374,7	1422,2	1441,8
0,97/	52,5	257,5	454,1	641,0	817,1	980,8	1130,8	1265,0	1381,6	1478,2	1552,2	1600,7	1620,4
0,98/	63,5	309,6	542,4	760,8	963,6	1149,4	1316,8	1464,4	1590,3	1692,8	1770,0	1819,6	1839,4
0,99/	78,2	378,3	657,8	915,7	1151,3	1363,5	1551,5	1714,2	1850,6	1959,5	2040,0	2090,8	2110,7
1,00/	98,0	470,0	810,0	1118,0	1394,0	1638,0	1850,0	2030,0	2178,0	2294,0	2378,0	2430,0	2450,0
1,01/	125,0	593,5	1012,6	1384,2	1710,3	1992,6	2232,9	2432,7	2593,6	2717,3	2805,0	2858,2	2878,3
1,02/	162,3	761,1	1284,2	1737,3	2125,7	2454,4	2727,9	2950,4	3125,8	3257,7	3349,3	3403,8	3424,0
1,03/	213,9	990,5	1651,2	2208,9	2675,1	3060,0	3372,7	3621,2	3812,7	3953,5	4049,3	4105,2	4125,5
1,04/	286,1	1306,3	2149,9	2842,5	3406,1	3859,4	4218,3	4496,7	4706,2	4856,8	4957,0	5014,2	5034,6
1,05/	387,4	1743,4	2831,3	3648,6	4384,4	4921,0	5334,7	5647,4	5877,0	6038,1	6143,0	6201,6	6222,2
1,06/	530,5	2351,3	3766,6	4860,7	5700,4	6338,6	6817,2	7169,3	7421,3	7593,9	7703,8	7763,8	7784,5
1,07/	733,2	3199,8	5055,3	6444,7	7478,4	8240,8	8796,3	9193,8	9470,8	9656,0	9771,0	9832,6	9853,3
1,08/	1021,4	4387,9	6836,8	8611,3	9890,1	10804,5	11451,2	11901,0	12206,0	12404,8	12525,3	12588,4	12609,3
1,09/	1432,1	6055,4	9305,8	11583,6	13172,4	14273,1	15028,0	15538,0	15874,4	16087,9	16214,3	16279,0	16299,9
1,10/	2018,3	8400,1	12734,7	15671,1	17652,5	18981,7	19865,3	20444,6	20816,0	21045,5	21178,1	21244,4	21265,5

8.1.5.
[cf. ch. 4.3.1]

```
                              * SOURCE LISTING *          A68   1.0.3   74330  12/12/74  16.20.05.

0001.          'BEGIN'                                                                              1
0002.            #COMPUTE ALPHA(A,D,Z) [C.F. 3.3.1]#                                                1
0003.            'STRING' TAB=" ";                                                                  1
0004.                                                                                               1
0005.            'PROC' F0=('INT' A,D,'REAL' Z) 'REAL';                                             1
0006.            'IF' 'ABS'(Z-1)<0.001                                                              2 DENOTATION
0007.              'THEN' (D+1)*(2*A+2-D)*0.5                                                       3 DENOTATION
0008.              'ELSE' ((Z**(A+2)-  Z**(A+1-D))/(1-Z)**2 + (D+1)/(1-Z))                          3
0009.            'FI';                                                                              1
0010.                                                                                               1
0011.            'PROC' DF0=('INT' A,D,'REAL' Z) 'REAL';                                            1
0012.            'IF' 'ABS'(Z-1) < 0.001                                                            2 DENOTATION
0013.              'THEN' ((A+2)*(A+1)*A - (A-D+1)*(A-D)*(A-D-1))/6                                 3 DENOTATION
0014.              'ELSE' ((A+2)*Z**(A+1) - (A-D+1)*Z**(A-D) -A*Z**(A+2) + (A-D-1 )*                4
0015.              Z**(A+1-D) + (D+1)*(1-Z))/(1-Z)**3                                               3 DENOTATION
0016.            'FI';                                                                              1
0017.                                                                                               1
0018.            'FOR' A 'FROM' 48 'TO' 48                                                          1 DENOTATION
0019.              'DO'                                                                             2
0020.                'INT' DELTA=A'OVER'12;                                                         2
0021.                'REAL' Z:=0.90;                                                                2
0022.                PRINT((NEW PAGE,"ALPHA(A,D,Z)",NEW LINE,"A=",WHOLE(A,0)));                     2
0023.                PRINT((NEW LINE,"D    "));                                                     2
0024.                'FOR'D 'FROM' 0 'BY' DELTA 'TO' A                                              2         TAG
0025.                  'DO' PRINT((TAB,WHOLE(D,-8))) 'OD';                                          2
0026.                PRINT((NEWLINE,"Z   /"));                                                      2
0027.                'WHILE' Z 'LE'1.101                                                            3 DENOTATION
0028.                  'DO'                                                                         3
0029.                    PRINT((NEW LINE,FIXED(Z,-4,2),"/"));                                       3
0030.                    'FOR' D 'FROM' 0 'BY' DELTA 'TO' A                                         3         TAG
0031.                      'DO'                                                                     4
0032.                        PRINT((TAB,FIXED((DF0(A,D,Z)/F0(A,D,Z))*Z -1/(Z+1),-8,1)))             4
0033.                      'OD';                                                                    3
0034.                    Z+:=0.01                                                                   3 DENOTATION
0035.                  'OD'                                                                         2
0036.              'OD'                                                                             1
0037.          'END'                                                                                0

PROGRAM LENGTH     000725B WORDS
```

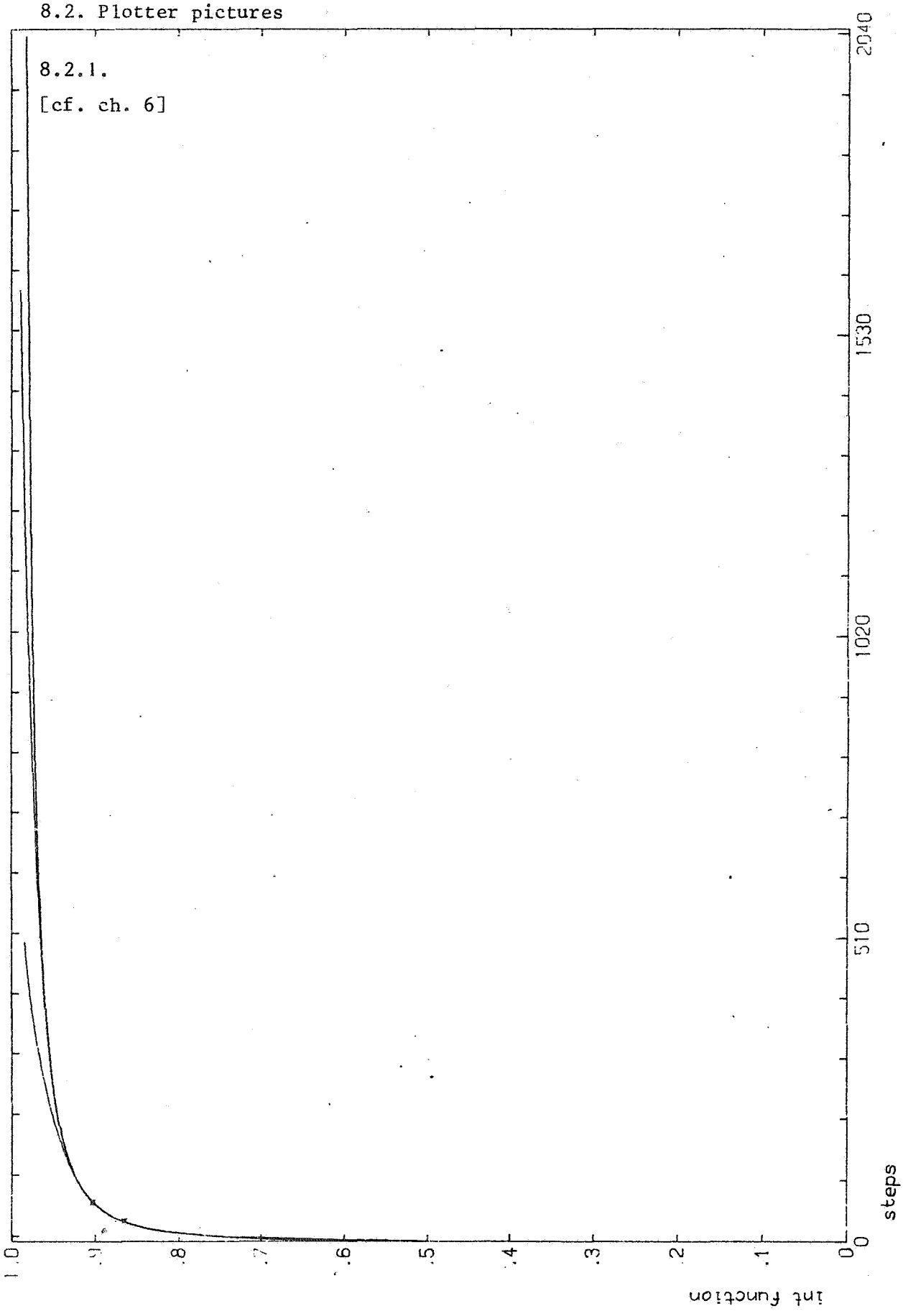
ALPHA(A,D,Z)
A=48

D / Z	0	4	8	12	16	20	24	28	32	36	40	44	48
0.90/	8.2	8.1	8.1	8.0	7.9	7.8	7.6	7.4	7.2	6.9	6.7	6.4	6.3
0.91/	9.1	9.0	8.9	8.8	8.7	8.5	8.3	8.1	7.8	7.5	7.3	7.0	6.9
0.92/	10.1	10.0	9.9	9.7	9.5	9.3	9.1	8.8	8.5	8.2	7.9	7.6	7.5
0.93/	11.3	11.2	11.0	10.8	10.5	10.2	9.9	9.6	9.3	8.9	8.6	8.3	8.2
0.94/	12.7	12.4	12.2	11.9	11.6	11.2	10.9	10.5	10.1	9.7	9.4	9.1	8.9
0.95/	14.2	13.9	13.5	13.1	12.8	12.3	11.9	11.5	11.0	10.6	10.2	9.9	9.8
0.96/	15.8	15.4	15.0	14.5	14.0	13.5	13.0	12.5	12.0	11.6	11.2	10.9	10.8
0.97/	17.6	17.1	16.5	15.9	15.4	14.8	14.2	13.7	13.1	12.7	12.2	11.9	11.8
0.98/	19.5	18.8	18.2	17.5	16.8	16.1	15.5	14.9	14.3	13.8	13.4	13.1	12.9
0.99/	21.5	20.7	19.9	19.1	18.3	17.5	16.8	16.2	15.6	15.0	14.6	14.3	14.2
1.00/	23.5	22.5	21.6	20.7	19.8	19.0	18.2	17.5	16.9	16.3	15.9	15.6	15.5
1.01/	25.5	24.4	23.3	22.3	21.3	20.4	19.6	18.9	18.2	17.7	17.3	17.0	16.9
1.02/	27.4	26.1	25.0	23.8	22.8	21.9	21.0	20.3	19.6	19.1	18.7	18.5	18.4
1.03/	29.2	27.8	26.6	25.4	24.3	23.3	22.5	21.7	21.1	20.6	20.2	20.0	19.9
1.04/	30.9	29.4	28.1	26.8	25.7	24.7	23.9	23.1	22.5	22.0	21.7	21.5	21.4
1.05/	32.5	30.9	29.5	28.2	27.1	26.1	25.2	24.5	23.9	23.5	23.2	23.0	22.9
1.06/	33.8	32.2	30.7	29.5	28.3	27.3	26.5	25.8	25.3	24.9	24.6	24.4	24.4
1.07/	35.1	33.4	31.9	30.6	29.5	28.5	27.8	27.1	26.6	26.3	26.0	25.9	25.8
1.08/	36.2	34.5	33.0	31.7	30.6	29.7	28.9	28.4	27.9	27.6	27.4	27.2	27.2
1.09/	37.1	35.4	33.9	32.6	31.6	30.7	30.0	29.5	29.1	28.8	28.6	28.5	28.5
1.10/	38.0	36.2	34.8	33.5	32.5	31.7	31.1	30.6	30.2	30.0	29.8	29.8	29.7

8.2. Plotter pictures

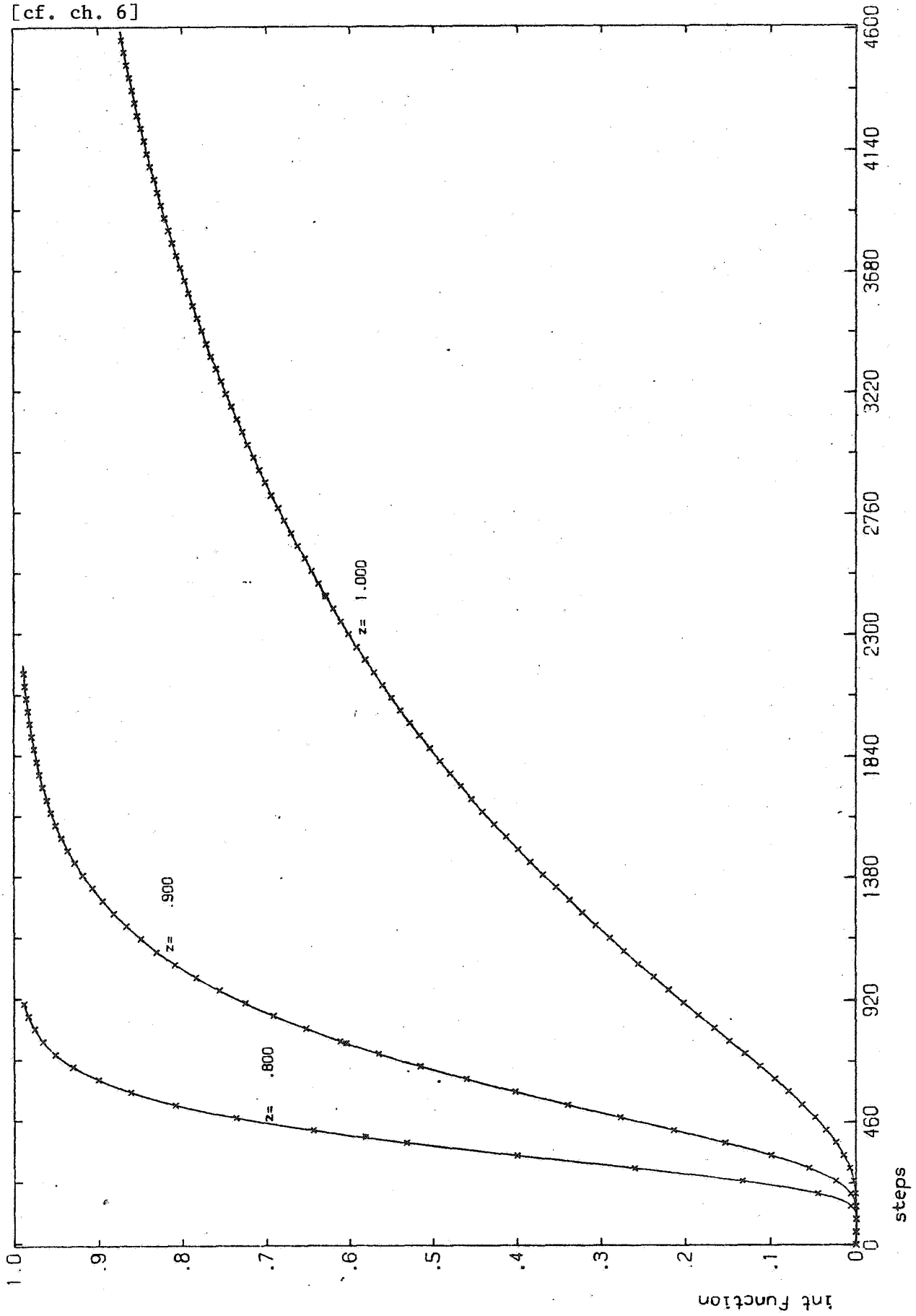
8.2.1.

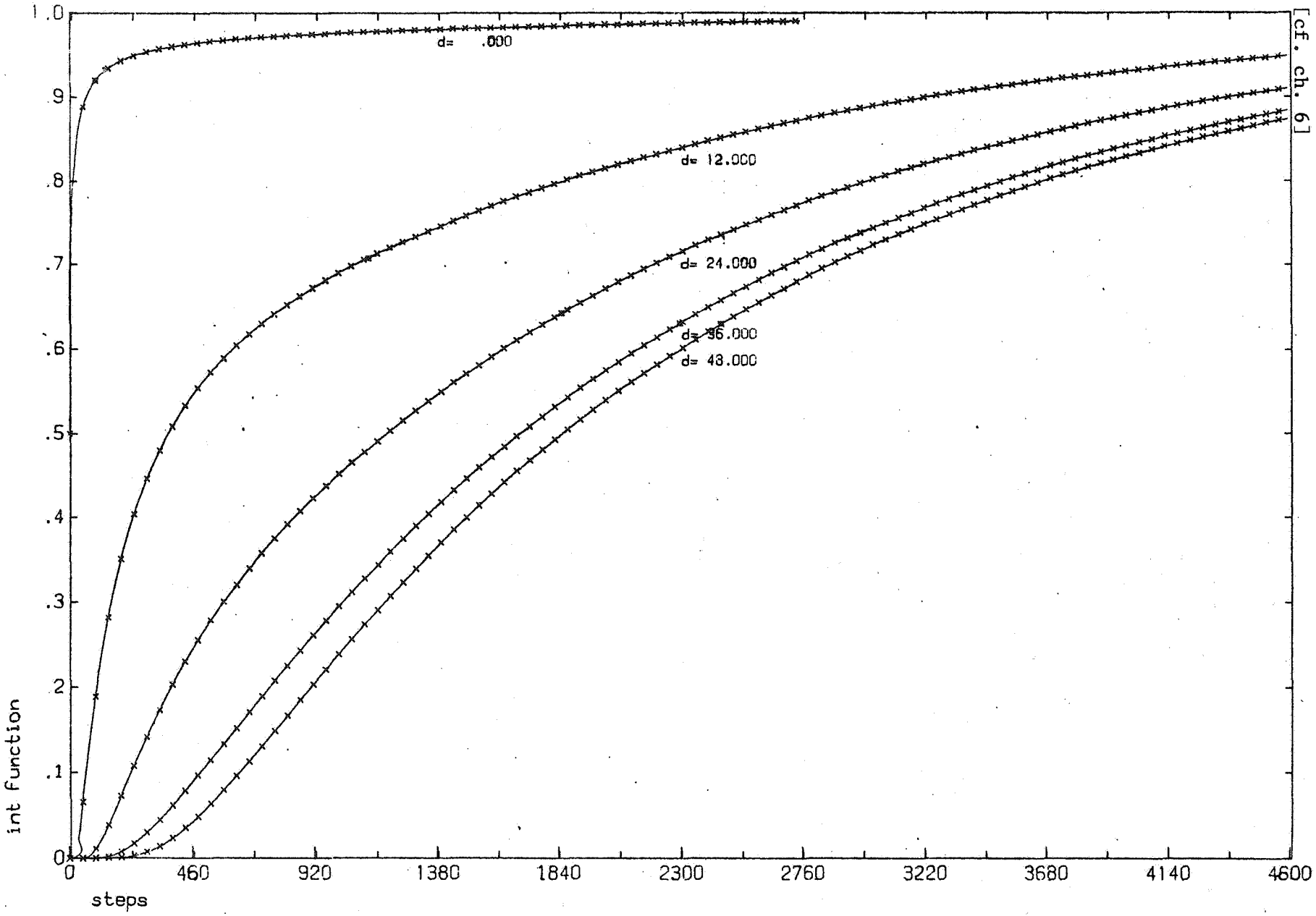
[cf. ch. 6]



8.2.2.

[cf. ch. 6]


int function
steps
1.0
.9
.8
.7
.6
.5
.4
.3
.2
.1
0
460
920
1380
1840
2300
2760
3220
3680
4140
4600
z= 1.000
z= .900
z= .800


1.0
.9
.8
.7
.6
.5
.4
.3
.2
.1
0
int function
d= .000
d= 12.000
d= 24.000
d= 36.000
d= 48.000
0
460
920
1380
1840
2300
2760
3220
3680
4140
4600
steps

9. REFERENCES

[1] COFFMAN, E.G. & A.C. McKELLAR, *On the Motion of an Unbounded, Markov Queue in Random Access Storage*, IEEE transactions on computers, June 1968, 600-603.

[2] SHEDLER, G.S., *A queuing model of a multi programmed computer with a two level storage system*, C-ACM 1611 (Jan. 1973), 3-10.

[3] KNUTH, D.E., *The art of computer programming* I.

[4] MORAN, P.A.P., *An introduction to probability theory.*

[5] FELLER, W., *An Introduction to Probability Theory and Its Applications* I.