Etude d'un algorithme d'ordonnancement en temps réel pour un monoprocesseur soumis à des contraintes de délais critiques

Daniel Julve

N° d'ordre :

Année 1976

THÈSE

présentée

devant L'UNIVERSITÉ PAUL SABATIER DE TOULOUSE (SCIENCES)

pour l'obtention

du Grade de DOCTEUR de Spécialité E.E.A. - Option "Automatique"

par

Daniel JULVÉ

Licencié ès Sciences Appliquées

ÉTUDE D'UN ALGORITHME D'ORDONNANCEMENT EN TEMPS RÉEL POUR UN MONOPROCESSEUR SOUMIS A DES CONTRAINTES DE DÉLAIS CRITIQUES

Soutenue le 15 Octobre 1976 devant la Commission d'Examen :

MM. G. GRATELOUP	Président
C. BETOURNÉ G. GIRALT H. H. JOHNSON J. AGUILAR-MARTIN	Examinateurs

Le travail présenté dans ce mémoire a été réalisé au "Department of Computing and Control, Imperial College" de l'Université de Londres et au Laboratoire d'Automatique et d'Analyse des Systèmes du CNRS.

Nous tenons à remercier Monsieur le Professeur J.H. Westcott, Directeur du "Department of Computing and Control" pour son accueil dans son Laboratoire.

Nous remercions également Monsieur le Professeur J. Lagasse, Directeur Scientifique au CNRS dont l'appui dans le cadre de la Convention d'échanges CNRS- Royal Society a rendu possible la réalisation de notre travail à Imperial College.

Nous remercions Monsieur le Professeur G. Grateloup, Professeur à l' INSA de Toulouse, qui a accepté la présidence du Jury de notre Thèse.

Notre reconnaissance à Monsieur H.H. Johnson, Senior Lecturer à Imperial College pour le grand nombre d'heures qu'il a consacrées à notre travail. Sa collaboration est à la base des résultats présentés dans notre mémoire.

Nous remercions Monsieur C. Bétourné, Maître de Conférences à l'Université Paul Sabatier pour les remarques qu'il a faites sur notre mémoire.

Nous remercions Monsieur G. Giralt, Directeur de Recherche au CNRS pour son appui à la réalisation de notre Thèse.

Nous remercions également Monsieur J. Aguilar-Martin, Maître de Recherche au CNRS pour son dévouement à éclaircir les aspects stochastiques de notre travail.

Nos remerciements à Madame A. Bartolomé pour son travail de dactylographie et tous ceux qui ont permis la réalisation de notre mémoire. Que tous y trouvent ma gratitude et ma reconnaissance sincères.

Finalement mes remerciements affectueux à mon épouse Patricia pour ses encouragements, son aide et les sacrifices qu'elle a accepté d'endurer au cours de la réalisation de notre travail.

ERRATA

page 11, 3ème paragraphe, 3ème ligne: Lire "k' = D(J_n,Q(t))" au lieu de "k' = D(J).

page 29 : Les expressions référenciées (4) et (5) doivent être référenciées (4') et (6).

page 39 : 1ère ligne: Lire " d'une classe de lois de commande acceptables pour la minimisation....." au lieu de "des lois de commande acceptables pour le critère de minimisation ....."

4ème paragraphe : Lire "Cet ensemble..." au lieu de "C'est ensemble....."

page 41, 1er paragraphe: Lire " à chaque instant" au lieu de "à l'instant".

3ème paragraphe et 4ème paragraphe sont remplacés par : " où $\mathcal{L}'_N$ est une loi de commande acceptable appartenant à $\mathcal{L}$, différente de la loi de commande optimale $\mathcal{L}^o_N$, et constituée des suites $\{u'_i\}$ de commandes U_i, dites grossières, qui ne considèrent pas le futur de $Q(t_i)$. Nous appellerons $\mathcal{L}^*_N$, la loi de commande du taux d'infaisabilité minimum, et $\mathcal{L}'_N$ une loi de commande grossière. "

page 42, 2ème paragraphe : Lire " date t_i " au lieu de "date t_N"

3ème paragraphe : Lire "il existe au moins une suite....." au lieu de "il existe une suite...."

4ème paragraphe : L'ennoncé du théorème 6 doit se lire "La loi de commande $\mathcal{L}^*_N$ du taux d'infaisabilité minimum est meilleure que toute loi $\mathcal{L}'_N$ grossière, différente de $\mathcal{L}^o_N$, pour la minimisation du critère C2."

5ème paragraphe : Dans l'expression (8') il faut lire

$$E\left[\sum_{i=o}^{N} L(u_i^*)/Q_o\right] \leqslant E\left[\sum_{i=o}^{N} L(u_i')/Q_o\right]$$

page 43, 2ème paragraphe : Lire " $\left\{ u_i^*/x_o \right\}_{i=o}^{N}$ " et" $\left\{ u_i' / x_o \right\}_{i=o}^{N}$ " et dans les expressions (8''), (14), (15), (16), (17), (18), (19), (20) et (22) lire u_i' au lieu de u_i.

page 45, 1er paragraphe : Lire "$E\left[L(u_n)/ x_n \right]$".

page 51, avant dernier paragraphe : Lire " meilleure que toute loi de commande grossière..." au lieu de "meilleure que toute autre loi de commande acceptable..."

page 52, 2ème paragraphe : Lire "commande $\mathcal{L}_N^*$ " et "commande $\mathcal{L}_N'$" successivement.

page 64, dernier paragraphe, avant-dernière ligne : Lire "est une commande grossière..." au lieu de "est une commande acceptable..."

page 66, 3ème paragraphe : Lire " C1 et C3" au lieu de"C1 et C2".

page 70, dernier paragraphe : Lire "sensibles par une commande sous-optimale..." au lieu de "sensibles d'une commande sous-optimale..."

page 74, 3ème paragraphe : Lire " (voir p. 29)"

page 75, 3ème paragraphe : Lire " valeurs 1 ou 2" au lieu de "valeurs 1, 2 ou 3"

To Patricia

Als meus pares, al meu
germà, als meus amics.

RESUME:

Cette étude présente la théorie et la conception d'un algorithme d'ordonnancement pour la gestion en temps réel sur un calculateur numérique des tâches soumises à des contraintes individuelles de délai d'achèvement.

Le système considéré ici est constitué par un monoprocesseur travaillant dans un environnement temps réel critique où les demandes de service ne peuvent pas être prises en compte immédiatement et doivent s'accumuler en file d'attente pour être satisfaites.

La solution proposée est basée sur une action de commande applicable chaque fois que l'arrivée d'une nouvelle tâche repousse la fin d'exécution d'au moins une tâche de la file d'attente accrue au-delà de son délai d'achèvement.

L'étude conduit à la réalisation d'un algorithme de commande sous-optimale pour la minimisation de l'espérance mathématique du nombre de tâches dont la demande d'exécution n'est pas satisfaite et dont les résultats sont vérifiés par simulation.

CHAPITRE 1

1.- Introduction

Le problème considéré ici est celui de l'ordonnancement dynamique des travaux en temps réel pour les systèmes informatiques où le respect des délais d'achèvement individuels pour chacune des tâches est pris comme une contrainte critique. [16]

Le système informatique considéré ici est constitué d'un processeur unique couplé à un processus extérieur tel que la réponse à chacune des demandes de traitement provenant du processus doive être complétée dans un délai défini à l'avance.

En général, un mécanisme d'interruption signale au Processeur la demande d'exécution d'une tâche qui, correctement complétée, permet le retour d'une réponse au processus.

Les délais d'achèvement critique ou absolus, qui ne peuvent être reculés sous aucun prétexte, caractérisent les processus dits de temps réel critique [16] alors que les délais relatifs où certains retards sont admissibles, caractérisent les processus dits de temps réel non critique.

Les systèmes informatiques destinés au contrôle des processus où les temps de réponse préfixés doivent être respectés, comme par exemple la commande des vaisseaux spaciaux [5] , constituent des exemples typiques d'une situation en temps réel critique.

Par contre, les systèmes informatiques du type exploitation en temps partagé, où l'interaction lente du processus constitué par

l'ensemble des utilisateurs, permet des retards acceptables pour chacune des demandes individuelles et qui ne dégradent en général pas le service rendu, constituent des exemples typiques de situations temps réel non critique.

Le problème de l'ordonnancement tel qu'il est posé ici, où le monoprocesseur joue le rôle d'un serveur unique, n'est pas exclusif du domaine de l'informatique et peut être rencontré dans un grand nombre d'applications où les demandes d'exécution de tâches données sont canalisées vers un ou plusieurs serveurs.

La théorie fondamentale est connue sous le nom de Théorie de l'Ordonnancement (Scheduling Theory, en anglais) [1], [2], [3], [4], [6], [13], [14], [16], [17].

Il est en général accepté que les performances des systèmes de service dépendent essentiellement de la stratégie d'ordonnancement des demandes. Etant donné que ces systèmes n'offrent pas tous le même type de service, il est logique que les critères de performances à satisfaire soient différents.

Par exemple dans un système d'exploitation en temps partagé, dans lequel normalement les utilisateurs payent pour l'exécution de leurs programmes, le critère de performance considéré est le débit en programmes traités par l'unité centrale qu'il s'agit de maximiser sous la contrainte d'un délai d'achèvement moyen acceptable [7].

Par contre, dans les situations de temps réel critique et en particulier dans les applications de contrôle des processus, les contraintes individuelles de délai d'achèvement doivent être respectées au moyen d'une stratégie d'ordonnancement appropriée. Cette même stratégie n'a aucune raison de satisfaire également les critères de performance de délai d'achèvement moyen acceptable.

Il est en conséquence raisonnable de s'attendre à ce que les stratégies d'ordonnancement soient différentes dans les deux cas.

D'une façon générale cependant, la littérature sur la théorie de l'ordonnancement semble s'intéresser très peu aux problèmes de temps réel critique. La plupart des travaux portent sur l'analyse statistique des performances des systèmes du type exploitation en temps partagé où, comme il a déjà été dit, la moyenne des délais d'achèvement doit être satisfaisante.

Ce manque d'intérêt pour le temps réel critique avait déjà été remarqué par Mc Naughton [13] en 1959 et par C.L. Liu et J.W. Layland [14] en 1973. Il est encore confirmé par les recherches de l'auteur, malgré le temps écoulé depuis 1959.

Les raisons de ce manque d'intérêt semblent être les suivantes:

Premièrement les disciplines d'ordonnancement les plus connues telles que "Premier Arrivé, Premier Servi", "Plus Petit d'abord" et "Temps Partagé " (respectivement connues par FCFS, SJFS, RR, de l'anglais "First Come First Served", "Shortest Job First" et "Round Robin") qui dans le cas le plus intéressant de l'ordonnancement dynamique sont les plus simples à manipuler, ne sont pas en général optimales pour le critère de satisfaction des délais d'achèvement individuels, ce qui n'est pas très surprenant vu que le paramètre délai d'achèvement n'y figure pas.

En fait, dans le cas de l'ordonnancement dynamique et pour des séquences d'arrivées aléatoires, l'utilisation de disciplines appropriées pour le respect des délais d'achèvement individuels complique beaucoup l'établissement de modèles mathématiques satisfaisants car les propriétés probalistiques des processus d'accumulation correspondants sont difficiles à établir (Voir Chapitre 3).

Deuxièmement, dans les situations de congestion en milieu temps réel critique, il est impératif d'introduire des critères de perte associés aux différents niveaux d'efficacité désirés [13].

Dans la pratique, en fait, une fonction de perte est quand même utilisée à la fois dans le cas de temps réel critique et non critique lorsque, en cas de saturation du système informatique, les nouvelles demandes de service sont refusées.

Cependant dans les situations de temps réel critique où un processeur unique assume la responsabilité de la commande automatique d'un processus dynamique, la ou les fonctions de perte doivent être définies d'une manière plus précise.

Ainsi, en prenant par exemple pour critère de perte une fonction du nombre des demandes de service pour lesquelles le délai d'achèvement individuel n'est pas respecté, la valeur finale de cette fonction sur un intervalle de temps donné ne doit pas dépasser une limite acceptable.

Là encore, très peu de travaux s'intéressent à l'étude d'ordonnancement où une fonction de perte est associée au non respect des délais d'achèvement individuels.

J.P. Gouyon [2], H.H. Johnson [3], [4], et R. Mc Naughton [13] considèrent tous implicitement une fonction de perte, mais le dernier uniquement en fournit une définition claire, dans [13], où les retards d'achèvement par rapport aux délais préfixés sont permis et le critère à minimiser est une fonction des retards encourus.

L'étude présentée ici par contre, est basée sur l'hypothèse que le traitement exécuté par le processeur est inutilisable s'il n'est pas terminé dans les délais individuels fixés pour chaque demande.

Comme dans les situations de congestion les demandes de service ne peuvent pas être immédiatement satisfaites, elles doivent être rangées dans une file d'attente avant d'accéder au contrôle de l'unité centrale.

Les retards ainsi causés peuvent alors repousser certaines dates de fin d'exécution au delà de leur délai d'achèvement.

Toute demande en tête de file pour laquelle cette situation se produit est alors ignorée par le processeur.

Le plan de travail de l'étude présentée ici est le suivant:

Le chapitre 2 fait l'objet de la formulation du problème, de l'analyse d'une file d'attente statique qui aboutit à l'optimalité de la discipline d'orientation pour le critère C1 de satisfaction des délais d'achèvement individuels.

Le chapitre 3 fait l'objet de l'étude des propriétés du processus d'accumulation constitué par une file d'attente orientée et définit le processus de faisabilité associé.

Le chapitre 4 présente l'étude d'une loi de commande sous-optimale pour la minimisation de la fonction de coût $C2 = E\left[C(0,t)\right]$.

Le chapitre 5 est consacré à l'étude des résultats expérimentaux fournis par une simulation en Fortran du modèle défini dans les chapitres précédents.

Le chapitre 6 fait l'objet d'une discussion sur les hypothèses faites, les lignes possibles de recherche comme continuation de l'étude présentée et la conclusion finale.

CHAPITRE 2

2. Analyse d'une file d'attente

Soit UC un processeur unique couplé à un processus temps réel critique P. Le processus P envoie à UC une séquence de demandes de traitement à satisfaire.

D'une manière générale, UC reconnaît les demandes à l'aide d'un mécanisme d'interruption et retourne les réponses appropriées à P sous forme de signaux de commande. Ce couplage entre UC et P est matérialisé par des interfaces spécialisés.

Dans les conditions de congestion pour UC, le problème que l'on propose de résoudre est celui de l'organisation de la séquence de demandes E de traitement de façon à satisfaire les deux critères C1 et C2.

C1.- Les délais d'achèvement individuels pré-définis de chaque demande sont respectés, si possible et

C2.- une fonction donnée du monbre de demandes ignorées est satisfaite.

Ainsi donc, le processeur et la séquence d'arrivées E peuvent être considérés comme un système dynamique S sur lequel est appliquéeune action de commande U -la stratégie d'ordonnancement- qui satisfait une fonction de coût donnée C -expression du nombre de demandes insatisfaites-.

La résolution du problème ainsi posé est basée sur la construction d'un modèle mathématique simple.

2.1.- Hypothèses sur la statique du modèle

L'ensemble des paramètres qui décrivent la structure du modèle cherché définissent ce que nous appellerons la statique. Ces paramètres sont relatifs au processeur UC et à l'ensemble des demandes (1) de traitement en provenance du processus P.

(1) Le sens du terme "demande" est ici identique à celui de "tâche" pour laquelle la demande est faite. Une tâche est en fait un programme de réponse à une demande donnée.

Pour l'ensemble T des demandes, les hypothèses faites sont les suivantes:

A1.- L'ensemble T des demandes est fini ou infini.

A2.- Chaque demande individuelle appartenant à T est définie par un identificateur J_n, une durée d'achèvement maximum RT, et un temps de service PT. L'indice n se rapporte à l'arrivée de la nième demande.

La variable RT représente l'intervalle de temps compris entre la date d'arrivée de la demande au processeur UC et le délai d'achèvement limite.

La variable PT représente la somme des intervalles de temps pendant lesquels la demande est traîtée par le processeur UC jusqu'à sa fin d'exécution.

La définition du temps de service PT est telle que les temps d'unité centrale nécessaires à la gestion des demandes peuvent être considérés comme négligeables ou bien inclus dans chaque PT.

A3.- Les demandes de service sont mutuellement indépendantes.

Ainsi l'exécution d'une demande quelconque n'est pas conditionnée par l'exécution d'une autre demande.

En ce qui concerne le processeur UC l'hypothèse faite ici est que:

A4.- Le processeur UC dispose d'une mémoire centrale infinie et d'un système d'interruptions pour la reconnaissance immédiate des demandes

Il est donc possible de mémoriser sans limitation d'espace toutes les demandes arrivant à l'UC et surtout d'avoir un accés immédiat à chacun des programmes de réponse associés à chaque demande.

En particulier A4 permet de s'affranchir de l'organisation dynamique de la mémoire en tant que contrainte d'ordonnancement. En

fait, du point de vue du respect des délais d'achèvement, toute limitation de la mémoire centrale se traduirait par des incréments des temps de service individuels PT correspondant à la pagination et autres échanges entre mémoire secondaire et mémoire centrale, comme il a déjà été envisagé en A2.

2.2.- Demandes acceptables et files d'attente infaisables

Soit t_n la date d'arrivée à UC de la nième demande J_n de la séquence E. Pour $n \leqslant m$, l'ensemble des demandes de E dont les dates d'arrivée sont comprises entre t_n et t_m inclus, constitue une sous-séquence de E que l'on notera $\mathcal{E}(t_n, t_m)$.

L'existence d'une séquence $\mathcal{E}(t_n, t_m)$ pour laquelle l'arrivée t_n se produit lorsque UC est encore occupé définit une situation de saturation pour UC dont la durée est l'intervalle (t_n, t_m) dit intervalle de saturation.

Or, un processeur unique tel que UC peut, à une date quelconque t, réaliser le traitement d'une seule demande uniquement et l'ensemble Q(t) des demandes en attente de fin d'exécution définit une file d'attente.

L'objet de ce paragraphe concerne l'étude des arrangements des éléments d'une file d'attente Q(t) à un instant donné quelconque t, de façon à déterminer les configurations de Q(t) qui satisfont le critère C1 défini en page 6

En accord avec les hypothèses faites en 2.1, chacune des demandes $J_n \in T$ est définie par une paire $J_n = (PT, FT)$ où les composantes $PT > 0$ et $FT > 0$ sont connues du processeur UC à la date d'arrivée[1] t_n et vérifient

$$RT = PT + FT$$

où FT est le temps libre intrinsèque.

(1)Le cas de demandes où PT est variable est traité en 2.4

Il est à remarquer qu'aucune des paires (PT,0) et (0,FT) PT $\geqslant$ 0, FT $\geqslant$ 0 n'appartient à T.

Définition 1

A une date donnée t, une file d'attente statique Q(t) est un ensemble ordonné de N(t) éléments $[k]$ (1) appartenant à T, k = 1, ..., N(t), en attente de fin d'exécution, où l'élément se trouvant en position 1, ou tête de file, est en cours de traitement par l'UC.

L'indice k définit la position d'un élément dans la file d'attente de sorte que $[k]$ est plus près de la tête de file que de $[k+1]$

A l'instant $t = t_n$ correspondant à la date d'arrivé de la nième demande de E, celle-ci est insérée dans la file d'attente en position k' (2) suivant une loi donnée D appelée discipline, telle que

(2) $k' = D(J_n, Q(t))$ $\qquad k' = 1, ..., N'(t)$

où N'(t) est le nombre d'éléments de la file d'attente accrue. La demande $[k']$ est définie par la paire

$[k'] = (PT_{k'}(t), FT_{k'}(t))$ où

(3) $PT(t)_{k'} = PT$

$FT(t)_{k'} = FT - SP(t)_{k'-1}$

avec $\qquad SP(t)_{k'-1} = \sum_{j=1}^{k'-1} PT(t)j$

ce qui entraîne pour la durée d'achèvement de $[k']$

(3') $\qquad RT(t)_{k'} = RT = PT + FT$

L'expression du temps libre $FT(t)_{k'}$ en (3) peut s'écrire

(1) L'indice entre crochets indique le contenu de la position correspondante.

(2) Les positions correspondant à une file d'attente accrue après l'insertion d'un nouvel élément seront notées ' .

(4) $$FT(t)_{k'} = RT(t)_{k'} - SP(t)_{k'}$$

De même à l'instant $t + dt$,

(4') $$FT\ (t+dt)_{k'+\ i'-1} = RT(t+dt)_{k'+\ i'-1} - SP(t+dt)_{k'+\ i'-1}$$

où i' est la position la plus proche de la tête de file telle que $0 < dt < SP_{i'}(t)$ de sorte que (4') devient

(4'') $$FT(t+dt)_{k'+\ i-1} = RT(t)_{k'+\ i-1}{}^{-dt} - SP(t)_{k'+\ i-1} + dt$$

c'est à dire

(5) $$FT(t+dt)_{k'+i-1} = FT(t)_{k'+i-1} \qquad , dt < t_{n+1} - t_n$$

d'où

Théorème 1 Pour toutes les demandes [k] , k=1,...., N(t) d'une file d'attente donnée Q(t), les temps libres $FT(t)_k$ restent constants entre deux arrivées successives t_n et t_{n+1} (1).

Une conséquence importante de ce théorème est:

Corollaire 1 Une condition nécessaire pour l'exécution satisfaisante d'un élément quelconque [k] d'une file d'attente statique est

(6) $$FT(t)_k > 0$$

Le fait que la condition (6) ne soit pas suffisante sera montré au chapitre 4, § 4.1

(1) L'UC est une machine séquentielle qui peut réaliser une seule opération à la fois. Les demandes simultanées sont donc prises en considération séquentiellement par UC.

Définition 2

A un instant donné t, tout élément [k] d'une file d'attente Q(t), k=1, ..., N(t) est dit acceptable si son temps libre $FT(t)_k$ est positif. Sinon il est dit inacceptable.

Définition 3

A un instant donné t, une file d'attente Q(t) est dite faisable si tous ses N(t) éléments sont acceptables. S'il existe au moins un élément [k] pour lequel $FT(t)_k \leqslant 0$, la file d'attente est dite infaisable.

D'après le Théorème 1, étant donnée une file d'attente faisable Q(t), à la date $t=t_n$ d'arrivée d'une nouvelle demande Jn en position k' = D(J), l'évaluation d'après (3) de tous les temps libres $FT(tn)_{i'}, i' \geqslant k'$ de la file d'attente accrue fournit un test pour la faisabilité de la nouvelle file d'attente $Q(t_n)$.

Comme par définition, seules les files d'attente faisables respectent les temps de réponse individuels, il est équivalent de dire qu'une file d'attente donnée Q(t) est faisable ou que le critère C1 est satisfait pour Q(t).

Le problème est maintenant de savoir s'il est possible, étant donné une file d'attente statique Q(t) infaisable, de réarranger ses N(t) éléments de façon à ce que Q(t) devienne faisable. Si ceci est possible, alors tous les éléments de Q(t) pourront être servis dans les délais individuels donnés et C1 est satisfait.

2.3.- Optimalité de la discipline d'orientation pour le critère C1 de faisabilité.

Le but de ce paragraphe est d'étudier les arragements combinatoires des N(t) éléments d'une file d'attente statique Q(t) et de retenir ceux pour lesquels les délais individuels des demandes sont respectés.

Ce problème a été exhaustivement étudié par H.H. Johnson qui a publié les résultats fondamentaux dans [4].

Dans ce qui suit, les théorèmes de base dus à H.H. Johnson sont donnés sans leur démonstration.

Définition 4

A un instant donné t, une file d'attente Q(t) est dite orientée si ses N(t) éléments sont tels que

(10) $RT_i(t) \leqslant RT_{i+1}(t)$ $\quad i = 1, \ldots, N(t)$

En appelant $d_k(t)$ le délai d'achèvement correspondant à l'élément $[k]$, la relation (10) est équivalente à

(10') $d_i(t) \leqslant d_{i+1}(t)$

puisque les délais et les temps de réponse sont liés par la relation $d_i(t) = RT_i(t) + t$, où t est la date courante d'observation.

L'ordonnancement défini par (10') est appelé SDFS (de l'anglais Shortest Deadline First Served) c'est à dire la discipline du "plus petit délai d'abord".

Par la suite il sera équivalent de dire qu'une file d'attente donnée Q(t) est orientée ou que ses éléments sont rangés suivant la discipline SDFS. Les deux notations utilisées seront $\mathcal{O}$ et SDFS respectivement.

La propriété remarquable de la discipline d'orientation $\mathcal{O}$ se traduit par les deux théorèmes suivant dont la démonstration peut être trouvée en (4).

Théorème 2

Les éléments de toute file d'attente faisable peuvent être arrangés de façon telle que la file d'attente résultante soit faisable et orientée.

Théorème 3

Si une file d'attente orientée donnée est infaisable, il n'est pas possible de réarranger ses éléments de façon telle que la file

d'attente résultante soit faisable.

Ces deux théorèmes peuvent être combinés en un seul

Théorème 4

La discipline d'orientation σ est optimale pour le critère C1 de faisabilité.

Une démonstration directe du théorème 4 peut être donnée comme suit:

Soit $t = t_n$ la date d'arrivée de la nième demande $J_n = (PT,FT)$ et soit $Q(t)$ la file d'attente faisable trouvée par J_n. D'après le Théorème 2 il est toujours possible de supposer que $Q(t)$ est orientée ce que nous noterons $Q_o(t)$, et faisable.

En raisonnant par l'absurde il sera prouvé que si l'insertion de J_n en position $i' = \sigma(J_n, Q\sigma(t))$ en utilisant la discipline d'orientation σ, cause l'infaisabilité de la file d'attente accrue $Q'\sigma(t)$, alors l'insertion de J_n en position $m' = D(J_n, Q_D(t))$ avec $m' \neq i'$, en utilisant toute autre discipline $D \neq \sigma$ causerait aussi l'infaisabilité de la file d'attente accrue $Q'_D(t)$.

Considérons pour celà les hypothèses H1 et H2 suivantes:

H1

Avec la discipline d'orientation σ, la demande $J_n = (PT,FT)$ est insérée en position $i' = i$ pour laquelle $RT = PT + FT$ vérifie:

$$(11) \quad \begin{cases} RT_{i-1}(t) \leqslant RT < RT_i(t) & i=2, \ldots, N(t) \\ 0 \leqslant RT < RT_1(t) & i=1 \\ RT_N(t) \leqslant RT & i=N(t) \end{cases}$$

et la file d'attente accrue $Q'_\sigma(t)$ est infaisable.

H2

En utilisant une discipline D différente de la discipline d'orientation σ, J_n est inséré en position $m' \neq i'$ et la file d'attente $Q'_D(t)$ accrue est faisable.

Sous l'hypothèse H1, l'infaisabilité de $Q'_{\sigma}(t)$ est dûe à l'une des trois possibilités:

H1. a.-

Seul le nouvel élément [i'] est inacceptable, ce qui s'écrit:

$$(12)\quad \begin{cases} FT < \sum_{j=1}^{i-1} PTj(t) & \text{et} \\ PT < FT_i^{\,m}(t) \end{cases}$$

où $FT_i^{\,m}(t)$ représente le plus petit des temps libres $FT_k(t)$ tels que $i \leqslant k \leqslant N(t)$, c'est à dire:

$$(12')\quad FT_i^{\,m}(t) = \min_{i \leqslant k \leqslant N(t)} (FT_k(t))$$

H1. b.-

Le nouvel élément J_n en position i' est acceptable, mais un ou plusieurs éléments de la file d'attente accrue deviennent inacceptables, ce qui se traduit par:

$$(13)\quad \begin{cases} FT > \sum_{j=1}^{i-1} PT_j(t) & \text{et} \\ PT \geqslant FT_i^{\,m}(t) & i \leqslant k \leqslant N(t) \end{cases}$$

H1. c.-

Le nouvel élément J_n en position i' et un ou plusieurs éléments de la file d'attente accrue sont innaceptables, ce qui se traduit par:

$$(14)\quad \begin{cases} FT \leqslant \sum_{j=1}^{i-1} PT_j(t) & \text{et} \\ PT \geqslant FT_i^{\,m}(t) & i \leqslant k \leqslant N(t) \end{cases}$$

Si H2 et H1.a sont toutes deux vraies, ceci entraîne qu'il existe une position m' = m dans la file d'attente orientée $Q_{\sigma}(t)$ telle que $1 \leqslant m \leqslant i-1$ et qui vérifie:

(15) $$\sum_{j=1}^{m-1} PT_j(t) < FT < \sum_{j=1}^{m} PT_j(t) \qquad \text{et}$$

$$PT < FT_i^{\,m}(t) \qquad m < i$$

Mais par définition de la position m

(16) $$PT < FT_m(t) \qquad m < i$$

ce qui entraîne

(16') $$PT + FT = RT < \sum_{j=1}^{m} PT_j(t) + FT_m(t) = RT_m(t)$$

et puique $RT_m(t) < RT_{i-1}(t)$, il vient finalement

(16'') $$RT < RT_m(t) < RT_{i-1}(t)$$

ce qui est contraire à (11) et à l'hypothèse H1.

Si H2 et H1. b sont toutes deux vraies, soit n la position la plus éloignée de la tête de file telle que $n = \max (i: PT \geqslant FT_i m(t))$, il existe alors une position $m' = m$ qui vérifie $i \leqslant n \leqslant m$ telle que:

(17) $$\begin{cases} FT > \sum_{j=1}^{m} PT_j(t) & \text{et} \\ FT_m(t) > PT > FT_i^{\,m}(t) \end{cases}$$

d'où l'on tire $RT = PT + FT > \sum_{j=1}^{m} PT_j(t) + FT_i^{\,m}(t)$

mais aussi

(17') $$RT > \sum_{j=1}^{n} PT_j(t) + FT_n(t) = RT_n(t)$$

et puisque $RT_n(t) \geqslant RT_i(t)$, on obtient finalement

(17'') $$RT > RT_n(t) \geqslant RT_i(t)$$

ce qui est contraire à (11) et à l'hypothèse H1.

Enfin, H2 et H1. c ne peuvent pas être toutes les deux vraies

puisque une discipline qui vérifirait à la fois

$$(18) \qquad FT > \sum_{j=1}^{i-1} PT_j(t) \qquad \text{et} \qquad PT < FT_i^{m}(t)$$

insèrerait la demande J_n dans une position m' telle que $m' \leq i-1$ et $m' > i$ à la fois ce qui est bien sûr impossible.

L'optimalité de la discipline d'orientation pour le critère de faisabilité C1 se trouve ainsi démontrée.

Une autre forme d'ennoncé pour l'optimalité de la discipline σ concerne l'étude des temps libres d'une file d'attente statique.

Théorème 5

Pour une file d'attente statique donnée $Q(t)$, la discipline qui maximise le temps libre minimum est celle d'orientation σ.

Démonstration

Avec les notations déjà utilisées, soit $Q_D(t)$ une file d'attente statique donnée, non orientée par le fait qu'il existe une seule position k telle que

$$(19) \qquad RT_k(t) > RT_{k+1}(t)$$

et soit $Q_\sigma(t)$ la file d'attente statique orientée correspondante. Alors $Q_\sigma(t)$ et $Q_D(t)$ diffèrent uniquement en ce que les éléments en position k et k+1 dans $Q_D(t)$ sont en position $\bar{k+1}$ et $\bar{k}$ respectivement dans $Q_o(t)$, où la barre indique l'ordonnancement correspondant à $Q_o(t)$.

Les deux files d'attente sont identiques pour les premières k-1 positions et pour les $N(t) - (k+1)$ dernières et il est évident que chacune des tâches correspondant à ces $N(t)-2$ éléments débute et est complétée à la même date exactement.

Pour toutes ces positions les temps libres $FT_j(t)$ sont donc rigoureusement les mêmes dans chacune des deux files.

En fait $Q_D(t)$ et $Q_\sigma(t)$ diffèrent en ce que les temps libres des éléments $[k]$ et $[\bar{k}]$ et $[k+1]$ et $[\bar{k}+1]$ respectivement sont différents.

Soit FT^m le plus petit des $N(t)-2$ temps libres $FT_j(t)$ identiques pour $Q_\sigma(t)$ et $Q_D(t)$.

Pour $Q_D(t)$ nous avons:

$$(19') \qquad FT_k(t) = RT_k(t) - \left(PT_k(t) + \sum_{j=1}^{k-1} PT_j(t)\right)$$

$$(19'') \qquad FT_{k+1}(t) = RT_{k+1}(t) - \left(PT_k(t) + PT_{k+1}(t) + \sum_{j=1}^{k-1} PT_j(t)\right)$$

ce qui entraîne

$$(20) \qquad FT_k(t) \geqslant FT_{\bar{k}+1}(t)$$

puisque $RT_{\bar{k}}(t) \geqslant RT_{\bar{k}+1}(t)$ et $PT_{\bar{k}+1}(t) > 0$

De même, pour $Q_\sigma(t)$ nous avons:

$$(21) \qquad FT_{\bar{k}+1}(t) = RT_{\bar{k}+1}(t) - \left(PT_{\bar{k}+1}(t) + PT_{\bar{k}}(t) + \sum_{j=1}^{\bar{k}-1} PT_j(t)\right)$$

$$(21') \qquad FT_{\bar{k}}(t) = RT_{\bar{k}}(t) - \left(PT_{\bar{k}}(t) + \sum_{j=1}^{\bar{k}-1} PT_j(t)\right)$$

où

$RT_{\bar{k}}(t) \leqslant RT_{\bar{k}+1}(t)$, $RT_k(t) = RT_{\bar{k}+1}(t)$,

$RT_{k+1}(t) = RT_{\bar{k}}(t)$, $PT_k(t) = PT_{\bar{k}+1}(t)$ et

$PT_{k+1}(t) = PT_{\bar{k}}(t)$

En comparant les expressions (19'') et (21), il vient

$$(22) \qquad FT_{\bar{k}+1}(t) \leqslant FT_{k+1}(t)$$

puisque $RT_{k+1}(t) = RT_{\bar{k}}(t) \leqslant RT_{\bar{k}+1}(t)$ et les sommes entre parenthèses sont les mêmes.

En comparant les expressions (21') et (19'') nous avons

(23) $$FT_{\bar{k}}(t) \geqslant FT_{k+1}(t)$$

puisque $PT_k(t) > 0$
les inégalités (20), (22) et (23) entraînent finalement

(24) $$\min_{D} (FT^m(t), FT_k(t), FT_{k+1}(t)) \geqslant \min_{\sigma} (FT^m(t), FT_{\bar{k}}(t), FT_{\bar{k}+1}(t))$$

ce qui établit le Théorème 5.

En conclusion, les Théorèmes 4 et 5 prouvent l'optimalité de la discipline d'orientation pour le critère de faisabilité C1.

En conséquence, la discipline d'orientation σ sera utilisée dans toute la suite de ce travail, sauf mention explicite.

Finalement, une propriété de la discipline σ concernant les priorités associées aux demandes d'exécution de tâches par l'UC peut être énnoncée comme suit:

Lemme 1

La discipline d'orientation est interruptible.

Une discipline est dite interruptible lorsqu'elle peut interrompre l'exécution de la demande en tête de file pour passer au traitement d'une autre demande.

Suivant cette définition, la discipline σ est nécessairement interruptible sinon un nouvel élément avec un temps de réponse $RT < RT_1(t)$ où $RT_1(t)$ est le temps de réponse à la date t de l'élément en position 1 (en cours d'exécution), serait inséré en position $i' \geqslant 2$ telle que $RT_1(t) > RT_{i'}(t) = RT$ ce qui contredit la définition de la discipline d'orientation σ.

Le caractère interruptible de la discipline d'orientation ne

doit donc pas être introduit en tant qu'hypothèse, comme c'est souvent le cas pour d'autres disciplines [15].

La seule hypothèse en fait concerne le type d'interruption utilisé qui est ici du type "avec reprise", où la tâche interrompue est reprise au dernier point d'interruption, lorsque son exécution n'est pas définitivement abandonnée par l'UC.

Dans le modèle considéré ici et d'un point de vue théorique au moins, l'optimalité de la discipline d'orientation implique qu'il n'est pas nécessaire d'associer des priorités externes aux demandes de traitement en temps réel.

En fait, dans les cas où la perte de demandes de traitement doit être envisagée, il est nécessaire d'introduire une partition de l'ensemble T des demandes en différentes classes. Cet aspect du problème est envisagé au chapitre 5.

2.4.- Cas des demandes à temps de service variable

Soit Q(t) une file d'attente donnée, faisable et orientée, et soient t_1^s et t_1^f respectivement les dates de début et de fin de traitement intrinsèque de l'élément [1] en tête de la file $Q(t_1^s)$.

Pour toute date t, $t_1^s \leq t < t_1^f$, $t \notin \bigcup_{i=1}^{q} \Delta\tau_i$ où $\Delta\tau_i$ sont les intervalles de temps pendant lesquels l'élément en tête de $Q(t_1^s)$ est interrompu, nous avons

$$PT_1(t) = PT_1(t_1^s) - (t - t_1^s) + h(t) \qquad (25)$$

On dira que l'élément [1] en t_1^s est à temps de service variable si $h(t) > 0$ et à temps de service non variable si $h(t) = 0$.

Cette définition entraîne $PT_1(t_1^f) > 0$ et $PT_1(t_1^f) = 0$ respectivement.

Dire que $PT_1(t)$ est variable peut alors être considéré équivalent à dire qu'il existe une sous séquence de demandes de E, commençant à la date t, $t_1^s \leq t' < t_1^f$, avec un temps d'inter-arrivée $\Delta = DT$ pour toutes les demandes de la sous séquence.

Si maintenant l'on suppose $J_n = (DT, DT)$ pour chacune de ces demandes, alors d'après (1) en 2. 2.

(26) $\qquad DT + DT = RT \leqslant RT_1(t')$

puisque par définition de DT, l'élément [1] en tête de file a une durée d'achèvement $RT_1(t')$ qui vérifie précisément (25).

Ainsi les demandes J_n de la séquence considérée sont "insérées" en position 1 aux dates t', t'+ DT, t'+ 2DT, t'+ 3DT, ...

La sous séquence de E s'arrête au moment où l'une de ses "demandes" cause l'infaisabilité de la file accrue ou bien lorsque l'arrivée d'une demande "vraie" avec une durée d'achèvement $RT < RT_1(t)$ interrompt le traitement de la demande à temps de service variable.

2.5.- Files d'attente infaisables avec pertes

Dans les paragraphes précédents il a été montré comment l'ensemble des demandes en attente d'exécution doit être ordonnée suivant la discipline d'orientation $\mathcal{O}$ de façon à satisfaire le critère C1 de faisabilité.

Ainsi, d'après l'optimalité de la discipline d'orientation, toute file d'attente statique Q(t) orientée et infaisable reste infaisable pour tout autre arrangement de ses éléments.

Dans les cas où l'arrivée d'une nouvelle demande transforme une file d'attente statique donnée, faisable et orientée, en une file d'attente infaisable, la question est alors de savoir ce qu'il faut faire des éléments devenus inacceptables dans la file accrue.

Dans son travail en référence [4], H.H. Johnson définit le test d'acceptation suivant pour toute nouvelle demande arrivant dans une file d'attente faisable:

Si l'insertion de la nouvelle demande J_n en position $i' = \mathcal{O}(J_n, Q(t)$

constitue une transition à un état infaisable de la file d'attente accrue, alors J_n est refusée.

Sinon J_n est acceptée et la file d'attente accrue reste faisable.

Pour la réalisation de ce test, il suffit d'évaluer les N'(t) temps libres $FT_{k'}(t)$ à la date $t=t_n$ d'arrivée en J_n (Voir (3) en 2.2)

Ainsi donc la réalisation de ce test d'acceptation entraîne qu'à tout instant donné t, une file d'attente Q(t) est, soit vide, soit faisable, mais toutes les demandes qui causent l'infaisabilité de la file accrue sont perdues.

La somme dans le temps de toutes les demandes refusées définit un coût cumulatif.

Une alternative possible à ce test d'acceptation consiste à permettre l'insertion de toute nouvelle demande J_n dans la file d'attente, même lorsque celle-ci résulte en une file accrue infaisable.

Les diverses façons suivant lesquelles la file d'attente accrue devient infaisable ont été analysées dans la démonstration du Théorème 4 en 2.3.

Par définition, il existe alors un élément au moins qui devient inacceptable, c'est à dire dont le temps libre correspondant devient négatif ou nul.

Soit $t = t_n$ la date d'arrivée de la nouvelle demande J_n et soit $D(t_n)$ l'ensemble ordonné par position croissante à partir de la tête de file, de tous les éléments inacceptables de la file d'attente accrue $Q'(t_n)$.

Par définition même, lorsque le premier élément de $D(t_n)$ atteint la tête de file on est sûr que son traitement sera complété après son délai d'achèvement.

En faisant l'hypothèse que chacune des demandes dont le traitement est complété après le délai correspondant est inutilisable, il est alors logique que l'UC n'entreprenne pas l'exécution d'une telle demande qui sera par conséquent ignorée.

Le reste des éléments de la file d'attente, s'il en reste, verront leurs temps libres respectifs augmentés de $PT_1(t)$. Cependant s'il existe encore des éléments appartenant à $D(t)$, pour lesquels cette augmentation du temps libre ne suffit pas à le rendre positif, alors la situation où un élément inacceptable atteint la tête de file se produira de nouveau et cet élément sera ignoré aussi par l'UC.

Plus précisément, à chaque instant $t = t_n$ où l'arrivée d'une demande J_n entraîne l'infaisabilité de la file d'attente accrue $Q'(t_n)$ il est possible d'associer une perte L_n définie comme suit:

Définition 5

A chaque instant $t = t_n$ d'arrivée d'un élément J_n, la perte $L_n = L\ (t=t_n)$ associée est le nombre d'éléments appartenant à $D(t)$ qui seront perdus lorsqu'ils atteindront la tête de file.

Pour toute demande J_n ne causant pas l'infaisabilité de la file d'attente accrue, alors $D(t) = \emptyset$ et $L_n = 0$.

Il est donc possible de déterminer au moment de l'insertion d'une nouvelle demande dans la file d'attente la perte L_n associée à cette insertion.

Considérons maintenant quelques propriétés de L_n.

Soit $Q(t)$ une file d'attente faisable qui devient infaisable lorsqu'elle est accrue d'une nouvelle demande J_n à l'instant $t = t_n$.

Soit $I(t)$ le nombre d'éléments de l'ensemble $D(t)$ des demandes inacceptables et $N'(t)$ le nombre d'éléments de la file d'attente accrue $Q'(t_n)$. La double inégalité suivante est satisfaite:

$$L_n \leqslant I(t) < N'(t) \tag{27}$$

Pour une file d'attente donnée faisable, la partie droite de l'inégalité traduit le fait que tous les éléments de la file d'attente accrue infaisable ne peuvent être tous inacceptables.

En effet, le nombre d'éléments inacceptable est maximum lorsque la nouvelle demande est insérée soit en position 1 avec $FT_1(t)=FT>0$ et tous les autres éléments sont inacceptables, soit en position 2 et tous les éléments en position $i \geqslant 2$ satisfont $FT_i(t) \leq 0$. Mais alors le nouvel élément en tête de file dont le temps libre est inchangé, reste acceptable.

La partie gauche de (27) traduit le fait que pour $I(t_n) > 1$, à moins que le premier élément de $D(t_n)$ lorsqu'il atteint la tête de file à la date $t_n + T$, $T > 0$ ne satisfasse

$$(28) \qquad PT_1(t_n+T) \geqslant \left|FT_j(t_n+T)\right|, \forall j \in D(t_n+T)$$

où $FT_j(t_n+T) \leqslant 0$

alors $L_n = I(t_n) > 1$

Lemme 2

La transition d'une file d'attente faisable orientée en une file d'attente infaisable orientée entraîne une perte $L_n \geqslant 1$, connue dès l'arrivée de la demande J_n correspondante.

En conclusion, étant donnée une file d'attente faisable orientée $Q(t)$ à chaque date $t = t_n$ d'arrivée d'une nouvelle demande J_n telle que la file d'attente ordonnée accrue soit infaisable, il existe trois possibilités:

a) Laisser la file d'attente accrue infaisable telle qu'elle est ; alors $L_n \geqslant 1$.

b) Réordonner la file d'attente accrue, mais d'après l'optimalité de la discipline d'orientation, la file d'attente résultante restera infaisable. En plus, comme la discipline d'orientation maximise le temps libre minimum -voir Théorème 5-, il se pourrait que par le réarrangement considéré, certains éléments acceptables deviennent inacceptables, d'où $L_n \geqslant 1$

c) Refuser l'insertion de la demande J_n qui cause l'infaisabilité et alors $L_n = 1$.

Le chapitre suivant à pour but d'étudier le problème de l'ordonnancement avec pertes en utilisant l'information fournie par la dynamique de la séquence d'arrivées E.

CHAPITRE 3

3.- Le processus d'accumulation Q(t)

Dans le chapitre précédent, le concept de perte a été introduit pour les situations de congestion où des demandes devenues inacceptables, atteignent la tête de file de Q(t).

Il a été montré comment à chaque date $t = t_n$ d'arrivée de la nième demande J_n, qui entraîne l'infaisabilité de la file d'attente accrue, est associée une perte L_n.

Une première question est de connaître la nature de la séquence des transitions de Q(t) auxquelles est associée une perte L_n et de se demander s'il n'est pas utile d'étudier la somme cumulée C(0,t) des pertes L_n sur un intervalle de temps donné.

Ainsi définie, C(0,t) constitue une variable de coût dont on se propose de minimiser l'espérance à partir de données probabilistiques concernant la séquence E d'arrivée de demandes à l'UC.

Cette information probabiliste est indispensable pour au moins deux raisons.

-D'abord sans cette information supplémentaire fournie par la dynamique de E, le problème étudié est déjà pourvu d'une solution qui est tout simplement celle du test d'acceptation de H.H. Johnson comme il a été montré à la fin du chapitre précédent. Cette solution ne peut être améliorée sans information supplémentaire.

-Ensuite l'établissement d'un modèle probabiliste pour la séquence E des arrivées de demandes permet d'étudier le comportement des stratégies d'ordonnancement devant les perturbations aléatoires sur E ce qui est souvent le cas dans les applications réelles de commande automatique de processus dynamiques.

Lorsque la séquence d'arrivée E présente des variations dans la régularité des arrivées(1) et/ou dans les temps de service et les

(1) Le cas où l'arrivée des demandes suit une loi périodique est étudiée en référence [14] et [18]

temps libres individuels, la file d'attente qui se crée dans des situations de congestion définit un processus stochastique appelé processus d'accumulation. [9],[18]

3.1.- Hypothèses sur la dynamique du modèle

Comme il vient d'être dit, toute quantité mesurable associée au processus d'accumulation considéré ici varie dans le temps autour d'une valeur moyenne.

En général,[19], un processus d'accumulation est défini par la donnée de la loi d'arrivée des demandes, la loi des temps de service et le mécanisme d'accumulation.

Cependant dans le modèle que l'on se propose d'établir ici, il faut ajouter la loi des durées d'achèvement ce qui introduit déjà une différence avec les modèles classiques utilisés en théorie des files d'attente.

-La Loi d'arrivée

La loi d'arrivée des demandes est définie par la séquence $\{t_n\}$, n=o, 1,... des dates d'arrivées successives. Il revient au même de considérer la suite des intervalles de temps Δ n entre arrivées où

(1) $$\Delta_{n+1} = t_{n+1} - t_n$$

La suite Δ n , n=0,1,... définit une séquence de réalisations d'une variable aléatoire Δ appartenant à une fonction de probabilité $F_1(\lambda, t)$ de paramètre λ telle que

(2) $$\mathbf{Pr}[\Delta \leqslant t] = F_1(\lambda, t)$$

-La Loi de service

Le temps de service pour chaque demande J_n est la composante PT de la paire J_n = (PT, FT) où PT est supposé être une variable aléatoire appartenant à une fonction de probabilité $F_2(\mu, s)$ de paramètre

telle que:

$$(3) \qquad Pr\ [PT \leqslant s] = F_2(\mu, s)$$

-La Loi des durées d'achèvement

La durée d'achèvement RT pour chaque demande J_n est supposée être une variable aléatoire.

Sachant que les variables aléatoires PT et RT intrinsèques sont liées par la relation RT = PT + FT, il suffit alors en connaissant la fonction de probabilité de PT de donner celle de RT ou bien celle de FT pour définir complètement la paire aléatoire J_n = (PT, FT).

On suppose ici que la fonction de probabilité des temps libres intrinsèques FT est donnée et que FT et PT sont indépendantes en probabilité et respectivement indépendants de Δ.

La fonction de probabilité $F_3(\rho, r)$, de paramètre ρ est telle que:

$$(4) \qquad P_r\ [FT \leqslant r] = F_3(\rho, r)$$

-Le mécanisme d'accumulation

Le mécanisme d'accumulation est défini par la loi d'ordonnancement qui fixe la séquence des éléments en attente de traitement.

Le choix de la discipline d'orientation Ó pour l'ordonnancement des éléments d'une file d'attente a été justifié au chapitre précédent.

On sait que cette discipline"classe" les éléments en attente de fin d'exécution par ordre non décroissant des durées d'achèvement $RT_i(t)$, $\forall t > 0$.

La connaissance des fonctions F_1, F_2, et F_3 et la discipline d'ordonnancement suffit à définir complètement le système dynamique constitué par la séquence E des arrivées et le monoprocesseur EC.

3.2.- Caractérisation du processus d'accumulation

Pour décrire complètement une file d'attente orientée Q(t) à tout instant donné t, il est nécessaire et suffisant de donner le nombre N(t) de ses éléments $[i]$ et les N(t) paires $[i] = [PT_i(t), FT_i(t)]$

Le fait que N(t) ne suffise pas à décrire la file d'attente Q(t) est une deuxième différence importante (Voir 3.1) avec les modèles classiques généralement utilisés dans la théorie des files d'attentes.

Plus précisément, définissons le processus d'accumulation $\{Q(t),\ t \geqslant 0\}$, continu dans le temps et dont l'état est donné par l'ensemble ordonné

$$(5) \qquad Q(t) = \{N(t),\ PT_1(t),\ FT_1(t),\ \ldots\ PT_N(t),\ FT_N(t)\}$$

L'expression (5) ci-dessus représente bien un état de Q(t), puisque l'addition de toute autre information pour décrire Q(t) serait redondante et la suppression d'au moins un terme de Q(t) ne suffirait pas à décrire le processus $\{Q(t),\ t \geqslant 0\}$ à une date t quelconque.

La discipline d'orientation utilisée ici définit l'insertion d'une nouvelle demande $J_n = (PT, FT)$ dans une des N'(t) positions de la file accrue, ce qui rend très difficile le calcul de la probabilité d'avoir un état Q'(t) à partir de la connaissance des états passés et présents.

Avec un choix approprié des fonctions F_1, F_2 et F_3 des variables aléatoires Δ , PT et FT, le processus d'accumulation Q(t) possède alors des propriétés simples.

Soient donc

$$(4)\quad \begin{cases} F_1(\lambda, t) = 1 - \exp(-\lambda t) \\ F_2(\mu, s) = 1 - \exp(-\mu t) \\ F_3(\rho, r) = 1 - \exp(-\rho r) \end{cases}$$

alors $\{Q(t),\ t \geqslant 0\}$ est un processus markovien puisque la probabilité d'avoir un état $Q(t+T)$ à partir de la connaissance des états passés et présent $Q(t)$ et $Q(s)$, $s<t$ ne dépend pas des états passés.

Si $\mathcal{E}$ est l'ensemble des états possibles de $Q(t)$, l'on a donc:

$$(5)\quad \Pr\left[Q(t+T) \in \mathcal{E} \,/\, Q(t),\ Q(s)\right] = \Pr\left[Q(t+T) \in \mathcal{E} \,/ Q(t)\right]$$

D'autre part, si les paramètres λ, μ et ρ en (4) sont invariants dans le temps, alors $Q(t)$ a une distribution stationnaire. Mais l'étude de l'évolution de $Q(t)$ dans le temps reste quand même difficile.

Cependant, d'après la définition de faisabilité donnée au chapitre 2, il suffit de considérer les temps libres $FT_i(t)$ pour connaître la faisabilité d'une file d'attente orientée $Q(t)$ à toute date t.

Nous allons ici étudier maintenant un processus associé à $Q(t)$ qui décrive les transitions entre les états faisables et infaisables de $Q(t)$.

En définissant pour chaque position i dans $Q(t)$ la fonction $Y_i(t)$

$$(7)\quad \begin{cases} Y_i(t) = 1 & \text{si } FT_i(t) \geqslant 0 \\ Y_i(t) = 0 & \text{si } FT_i(t) \leqslant 0 \end{cases}$$

et la fonction $Z(t)$ associée à l'ensemble des $N(t)$ éléments de $Q(t)$:

$$(8)\quad Z(t) = \prod_{i=1}^{N(t)} Y_i(t)$$

on obtient alors

Lemme 3: Quelle que soit la date t, Q(t) est faisable (infaisable) si et seulement si $Z(t) = 1 (= 0)$

A partir de Z(t) et des intervalles d'infaisabilité nous allons construire un processus stochastique, plus simple que Q(t).

3.3.- L'ensemble des intervalles d'infaisabilité

Etant donné une file d'attente faisable Q(t) et une nouvelle demande J_n arrivant à la date $t = t_n$, la file d'attente accrue Q'(t) devient infaisable s'il existe une position i au moins de Q(t) telle que l'événement

$$(10) \quad \left\{ RT_i(t) < RT \leqslant RT_{i+1}(t) \quad \text{et} \quad (PT < FT_i^m(t) \quad \text{et/ou} \quad FT \leqslant SP_i(t) \right\}$$

est certain.

Où PT et FT sont les composantes de la paire aléatoire J_n et où

$$FT_i^m(t) = \min_{k \geq i} FT_k(t) \quad ; \quad SP_i(t) = \sum_{j=1}^{i} PT_j(t) \quad \text{et}$$

$$RT_i(t) = SP_i(t) + FT_i(t)$$

Pour chaque position i' dans la file d'attente accrue Q'(t) définisons les intervalles $I_{i',1}(t)$, $I_{i',2}(t)$, $I_{i',3}(t)$

$$(11) \quad \begin{cases} I_{i',1}(t) = \left]RT_i(t) \ , \ RT_{i+1}(t)\right] \\ I_{i',2}(t) = \left]FT_i^m(t) \ , \infty \right[\\ I_{i',3}(t) = \left]0 \ , \ SP_i(t)\right] \end{cases}$$

L'ensemble d'infaisabilité de $I_{i'}(t)$ associé à la position i' est défini par l'ensemble des trois intervalles de R ci-dessus.

(12) $I_{i'}(t) = \{I_{i',1}(t) , I_{i',2}(t) , I_{i',3}(t)\}$

Pour i' = 1, les expressions en (11) s'écrivent:

(11') $\begin{cases} I_{i',1}(t) =]0 , RT_1(t)] \\ I_{i',2}(t) =]FT_1^m(t), \infty[\\ I_{i',3}(t) =]0, 0[\end{cases}$

et pour i' = N(t)

(11'') $\begin{cases} I_{N1}(t) =]RT_N(t), \infty[\\ I_{N2}(t) =]\infty, \infty[\\ I_{N3}(t) =]0, SP_N(t)] \end{cases}$

Soit maintenant A l'évènement défini par

(12) A={ à la date $t=t_n$ d'arrivée de la nième demande J_n, Q'(t) est infaisable }

La réalisation de A implique celle de l'évènement $A_{i'}(t)$, i'=1, ..., N'(t) défini par:

(12') $A_{i'}(t) = \{$ à la date $t=t_n$ d'arrivée de J_n, la terne (RT, PT, FT) $\in$ à l'ensemble $I_{i'}(t)\}$

C'est à dire qu'il existe une position i' dans la file accrue telle que l'évènement

(13) $\{RT \in I_{i',1}(t)$ et $(PT \in I_{i',2}(t)$ et/ou $FT \in I_{i',3}(t))\}$

est certain.

De sorte que les évènement A et $A_{i'}(t)$ en (12) et (12') sont liés par la relation:

$$(14) \qquad A = \bigcup_{j=1}^{N'(t)} A_j(t)$$

Mais par construction même des ensembles d'infaisabilité $I_{i'}(t)$ nous avons, pour $i \neq j$, $I_i(t) \neq I_j(t)$ et donc

$$(14') \qquad A_i(t) \cap A_j(t) = \emptyset$$

La probabilité de réalisation de l'évènement A peut alors s'écrire en vertu de (14) et (14')

$$(15) \qquad P_r[A] = \sum_{j=1}^{N'(t)} P_r[A_j(t)]$$

Par extension de la définition de demande J_n donnée au Chapitre 2, l'évènement en (13) ci-dessus sera noté symboliquement $J_n \in I_{i'}(t)$ où $J_n \equiv (RT, PT, FT)$.

Alors l'évènement $\{J_n \in I_1(t) \cup J_n \in I_2(t) \ldots\ldots \vee J_n \in I_N(t)\}$ sera noté, toujours de façon symbolique.

$$(16) \qquad J_n \in I(t) = \{J_n \in I_1(t) \quad \ldots \qquad \ldots \quad J_n \in I_N(t)\}$$

où $I(t)$ est l'ensemble d'infaisabilité associé à la file $Q(t)$.

3.4.- Le processus de faisabilité

D'après sa définition en (8), $Z(t)$ est une fonction déterministe de $Q(t)$.

(16) $\quad Z(t) = \Psi(Q(t))$

où le processus d'accumulation lui-même est décrit par

(17) $\quad Q(t+dt) = \varphi(Q(t)) + W_t$

W_t est la perturbation aléatoire correspondant à l'arrivée d'une demande J_n dans l'intervalle $(t, t+dt)$ et φ est une fonction non croissante des composantes de $Q(t)$.

Considérons maintenant la probabilité d'avoir une file d'attente infaisable à la date $t + dt$, étant donné une file $Q(t)$, c'est à dire:

(18) $\quad P_r \left[Q(t+dt) \text{ infaisable } / \ Q(t) \right]$

que nous pouvons écrire

(19) $\quad P_r \left[Z(t+T) = 0 \ / \ Q(t) \right]$

ou encore

(19') $\quad P_r \left[Z(t+T) = 0 \ / \ Q(t), \ Z(t) \right]$

puisque étant donné $Q(t)$, la connaissance de $Z(t)$ n'apporte aucune information supplémentaire nécessaire à la détermination de $Z(t + dt)/Q(t)$.

Donc à la date t, l'état futur de $Z(t + dt)/ Q(t)$ est conditionellement indépendant de $Z(t)$.

Ceci n'entraîne pas que le processus constitué par la suite des $\{Z(t),\ t \geqslant 0\}$ soit une séquence indépendante, car la connaissance de $Q(t)$ en (19) entraîne celle de $Z(t)$.

Le processus $\{Z(t),\ t \geqslant 0\}$ représente bien la séquence des états faisables / infaisables de $Q(t)$, mais sans autre information supplémentaire il n'est pas possible de calculer simplement les probabilités de transition:

$$(20) \qquad P_r\left[Z(t+dt) = j \ / \ Z(t) = i\right] \qquad i = j = 0,\ 1$$

Par contre, au moyen des intervalles d'infaisabilité introduits au paragraphe précédent, il est possible de calculer la probabilité en (19).

En effet, en utilisant les notations symboliques définies en 3.3 , nous pouvons écrire:

$$(21) \qquad P_r\left[Z(t+dt)=0 \ / \ Q(t)\right] = P_r\left[\Delta \leqslant dt \text{ et } Q(t+dt) \neq \emptyset \text{ et } J_n \in I(t+dt) \ / \ Q(t)\right]$$

où la variable aléatoire Δ est indépendante en probabilité de $Q(t)$, $Q(t+dt)$ et de J_n, de sorte que l'expression ci-dessus devient:

$$(22) \qquad P_r\left[Z(t+dt)=0 \ / \ Q(t)\right] = P_r\left[\Delta \leqslant dt\right] . P_r\left[Q(t+dt) \neq \emptyset \text{ et } J_n \in I\ (t+dt) / \ Q(t)\right]$$

Au paragraphe 2.2 du chapitre 2, il a été précisé qu'il n'existe pas de demandes $J_n = (0,\ 0) \in T$. Donc la réalisation de l'évènement $J_n \in I(t+dt)$ implique celle de $Q\ (t+dt) \neq \emptyset$ d'où

$$(23) \qquad P_r\left[Z(t+dt)=0 \ / \ Q(t)\right] = P_r\left[\Delta \leqslant dt\right] . P_r\left[J_n \in I(t+dt) \ / \ Q(t)\right]$$

En prenant pour fonction de répartition de la variable aléatoire Δ , l'expression $F_1(\lambda, t)$ donnée en (4), l'expression devient:

$$(24) \qquad P_r\left[Z(t+dt) = 0 \ / \ Q(t)\right] = \lambda . Pr\left[J_n \in I(t+dt)/Q(t)\right] . dt$$

qui satisfait bien la condition

$$(25) \qquad Pr\left[Z(t+dt) = 0 \ / \ Q(t) = \emptyset\right] = 0$$

puisque l'ensemble d'infaisabilité $I(t+dt)$ pour $Q(t+dt) = \emptyset$ est composé d'intervalles $]0, 0]$ et la partie droite de (24) est bien égale à 0.

Lorsque $Q(t)$ évolue dans le temps, la suite des variables aléatoires $\{ Z(t+dt) / Q(t) \}$ définit un processus stochastique que nous noterons $\{F(t+T), t \geqslant 0\}$ et que nous appellerons le processus de faisabilité associé à $Q(t)$.

En reprenant l'expression (24) plus haut, nous définissons maintenant le taux d'infaisabilité du processus $\{F(t)\}$ par l'expression $q_o(t)$.

$$(26) \qquad q_o(t) = \lambda \cdot P_r \left[J_n \in I(t+dt) / Q(t) \right]$$

de sorte que pour les transitions de $F(t)$ à l'état 0, nous avons:

$$(27) \qquad P_r \left[F(t+dt) = 0 \right] = q_o(t) \cdot dt$$

avec, bien sûr, $q_1(t) = 1 - q_o(t)$.

Ainsi donc, au processus d'accumulation $\{Q(t), t \geqslant 0\}$ est associé le processus $\{Z(t)\}$, fonction déterministe de $Q(t)$, qui décrit les états faisables/infaisables instantanés de $Q(t)$.

Les états futurs de $Z(t)$, connaissant $Q(t)$ sont alors représentés par le processus de faisabilité $\{F(t+dt), t \geqslant 0\}$ où la probabelité d'avoir $F(t+dt) = i$, $i = 0, 1$, est précisément donnée par (24).

CHAPITRE 4

4.- Etude de la commande du processus de faisabilité

Dans le chapitre 3, l'introduction de données probabilistes pour le modèle de file d'attente considéré a conduit à la carectérisation du processus d'accumulation $\{Q(t),\ t \geqslant 0\}$ du processus de faisabilité associé $\{F(t),\ t \geqslant 0\}$ et à l'expression explicite du taux d'infaisabilité $q_o(t)$.

A l'aide de ces résultats et en précisant les notions de perte introduites dans les chapitres 1 et 2, on se propose maintenant de faire une étude des possibles lois de commande qui permettent de considérer la minimisation de l'espérance du nombre $C(0,t)$ des demandes ignorées.

Les commandes considérées dans les paragraphes qui suivent consistent justement à déterminer les demandes qui doivent être ignorées par le processeur et ceci à chaque transition de $Z(t)$ à l'état 0.

En reprenant les considérations faites au chapitre 2 sur la perte L_n à chaque changement d'état de $Z(t)$, il est clair qu'il existe plusieurs manières de rejeter des éléments de la file d'attente accrue.

Dans les paragraphes qui suivent les hypothèses suivantes sont faites pour l'étude de la commande de $F(t)$.

H.3.- La file $Q(t)$ doit être faisable à toute date t, sauf peut-être aux dates de transition t_n. Comme condition limite on admettra qu'une file $Q(t)$ vide est faisable.

H.4.- La loi de commande cherchée est constituée d'une suite de commandes u_n applicables aux dates de transition de $Z(t=t_n)$ à l'état 0 uniquement et qui consiste à déterminer les demandes qui doivent être ignorées par le processeur.

H.5.- La perte L_n associée à chaque transition de F(t) à l'état 0 est imposée égale à 1.

Il sera admis par la suite pour que la notion d'état de Z(t) garde son sens, que l'état Z(t) = i ne peut changer qu'à l'arrivée d'une nouvelle demande. C'est à dire que si Z(t) = 0 après l'arrivée d'une demande et si l'action de commande u_n sur Q'(t), laisse la file commandée dans un état faisable forcé, celle-ci sera considérée quand même infaisable jusqu'à la prochaine arrivée, bien que pour l'UC, $Q'(t, u_n)$ soit de nouveau faisable.

Comme il a été dit, l'action de commande cherchée consiste à déterminer l'élément à rejeter de la file accrue infaisable mais l'action elle-même de rejet peut avoir lieu lorsque la demande choisie atteint la tête de file où elle est alors ignorée par le processeur.

4.1.- Commandabilité du processus de faisabilité

Soit Q(t) une file d'attente faisable orientée et non vide à la date $t=t_n$ d'arrivée de la nième demande J_n.

Il a été montré en 3.2 que si la demande J_n appartient à l'ensemble d'infaisabilité I(i), la file d'attente Q'(t) accrue par insertion de J_n en position $i'=\sigma(J_n,Q(t))$ est infaisable.

L'ensemble D(t) de tous les éléments $[k']$ inacceptables de Q'(t) a été introduit en 2.4.

Soit i'_o la position en Q'(t) de l'élément de D(t) le plus près de la tête de file. D'après 2.4 l'on sait que $i'_o > 1$.

Soit V(t), l'ensemble des éléments $[j']$, $j' \leqslant i'_o$ de la file accrue Q'(t) dont le temps de service $PT_{j'}(t)$ est supérieur au plus grand temps libre négatif de $[k'] \in D(t)$ ce qui s'écrit

$$(1) \qquad PT_{j'}(t) > \max_{k' \in D(t)} \left| FT_{k'}(t) \right| \qquad 1 \leqslant j' \leqslant i'_o$$

V(t) n'est jamais vide puisque au moins le nouvel élément J_n en position i' lui appartient, de sorte que si p(t) est le nombre d'éléments de V(t), l'on a $p(t) \geqslant 1$.

Il est à noter que dans le cas où le nouvel élément J_n lui-même est inacceptable, alors $i' = i'_o$.

Ainsi donc l'action de rejeter $m(t) \leqslant p(t)$ éléments de V(t) à chaque transition de Z(t) à l'état 0, définit une commande pas à pas qui satisfait le critère C3 dit de faisabilité:

$$C3 = \left\{ Q(t) \text{ est faisable } \forall\, t \neq t_n \,,\ n=0,.... \right\}$$

(2)

$$= \left\{ Z(t) = 1 \,,\ \forall\ \ t \neq t_n \,, \qquad n=0,.... \right\}$$

Une telle loi existe toujours puisque pour m(t) = 1, on connaît déjà la commande d'acceptation de H.H. Johnson qui satisfait bien C3.

D'après sa définition, l'ensemble V(t) contient uniquement des éléments acceptables de Q'(t), sauf lorsque le nouvel élément J_n en position $i' = i'_o$ est lui aussi inacceptable.

Par conséquent, la condition (6) du chapitre 2 n'est pas suffisante, comme cela avait été annoncé.

Cette loi de commande qui rejette à chaque transition de Z(t) à l'état 0, 1 (ou plusieurs) éléments de V(t), fournit en principe une valeur pour le critère $C2 = E\left[C(0,t)\right]$, qui n'a aucune raison d'être optimale pour la minimisation de ce critère.

Nous dirons qu'une telle loi est acceptable pour la minimisation de C2.

Sachant que l'ensemble des lois de commande acceptables pour min C2 n'est pas vide, nous nous proposons maintenant d'améliorer la

performance des lois de commande acceptables pour le critère de minimisation de C2, compte tenu des hypothèses H3, H4 et H5 ci-dessus.

Cette recherche d'une loi de commande sous optimale et non pas optimale est dûe au fait que ni les transitions de $Q(t)$ aux états futurs ni celles de $Z(t)$ aux états futurs ne sont connues et sont difficiles à déterminer.

Ainsi donc, à la date d'arrivée $t = t_n$ de la même demande J_n soit $u(t=t_n)$ la commande à appliquer à $Q(t)$ que nous noterons

(3) $$u^{j'}_n = \left\{ \text{A la date } t=t_n, \text{ rejeter l'élément } \left[j'\right] \in V(t_n) \right\}$$

et U_n l'ensemble des commandes acceptables $u^{j'}_n$ est défini par

(4) $$U_n = \left\{ u^{j'}_n \quad .t \ .q \left[j'\right] \quad V(t_n) \right\}$$

C'est ensemble n'est jamais vide, puisque $V(t_n)$ ne l'est jamais non plus.

Mais toute action de rejet u^j_n modifie la file d'attente $Q'(t_n)$ qui devient $Q\ (t, u_n)$ et si la longueur de Q en temps de service des éléments de $Q\ (t, u_n)$ est telle que $Q > dt$, $dt > 0$, alors les ensembles d'infaisabilité $I(t+dt)$ qui sont bien sûr des fonctions de $Q'(t_n+dt)$ deviennent en plus des fonctions de la commande u^j_n que l'on écrira $I(t\ \ dt, u^j_n)$.

Or d'après les expressions (24) et (25) du chapitre précédent le taux d'infaisabilité $q_o(t)$ est fonction de $I(t+dt, u^j_n)$ ce qui entraîne

(5) $$q_o(t_n) = q_o(u^j_n)$$

et la commandabilité du taux d'infaisabilité $q_o(t)$ pour le critère min C2 est établie.

Nous dirons que q_o est F-commandable pour min C2 et que les commandes $u_n^j \in U_n$ sont acceptables pour ce critère.

La dynamique des processus $Q(t)$, $Z(t)$ et $F(t)$ alors décrite par les équations

$$(6) \quad \begin{cases} Q(t_n + dt) = \xi\,(Q(t),\ u^j_n) + \quad w_t \\ Z(t_n) = \Psi\,(Q(t_n)) \\ F(t_n + T) = Z(t_n + T) \ /\ Q(t,\ u^J_n) \end{cases}$$

4.2.- Etude de la commande du processus de faisabilité pour le critère C2 = E [C(0,t)].

Au paragraphe précédent il a été montré que sous les hypothèses H.3, H.4 et H.5, U_n est l'ensemble des commandes acceptables à l'instant t_n donc que le processus $\{F(t+T),\ t \geqslant 0\}$ est commandable pour C2.

Ainsi donc, toute suite $\mathcal{L}_N$ telle que

$$\mathcal{L}_N = \{u_o, \ldots\ldots, u_N\} \tag{7}$$

Où chaque u_i est une commande appartenant à l'ensemble des commandes acceptables $U_i = U\ (t=t_i)$ definit une loi de commande acceptable pour la minimisation de C2.

En appelant $\mathcal{L}$ l'ensemble des suites $\mathcal{L}_N$ en (7), nous nous proposons de montrer que la loi de commande $\mathcal{L}_N \in \mathcal{L}$ construite à partir de l'ensemble des commandes u_i qui à chaque transition de $Z(t_i)$ à l'état 0 minimisent le taux d'infaisabilité $q_o(u_i)$ est une loi de command sous-optimale pour min C2 qui vérifie

$$E\left[C(0,t_N,\mathcal{L}_N^*)\right] \leqslant E\left[C(0,\ t_N,\mathcal{L}_N)\right] \tag{8}$$

où $\mathcal{L}_N$ est une loi de commande acceptable appartenant à $\mathcal{L}$ différente de la loi de commande optimale $\mathcal{L}_N^o$.

Nous appellerons $\mathcal{L}_N^*$, la loi de commande du taux d'infaisabilité minimum.

D'après le chapitre 3 et compte tenu des hypothèses H.3, H.4 et H.5 du paragraphe 4.1, nous savons que le coût L_i à chaque transition de $Z(t_i)$ à l'état 0 est une variable aléatoire qui peut prendre les valeurs 0 et 1 de sorte que l'espérance du coût $E\left[L_{i+1}/Q(t_i)\right]$ s'écrit

$$E\left[L_{i+1}\ /\ Q(t_i)\right] = \alpha(t_{i+1}).q_o(t_{i+1}) \tag{9}$$

où $\alpha(t_{i+1})$ est la probabilité d'avoir une arrivée à la date t_{i+1}.

A chaque date t_N de transition de $Z(t_i)$ à l'état 0, la commande $u_i \in U_i$ qui minimise le taux d'infaisabilité $q_o(t_{i+1}, u_i)$ minimise l'espérance du coût à l'instant t_{i+1} et nous pouvons écrire, avec $Q(t_i) = Q_i$

$$(10) \qquad E\left[L(u_i) \,/\, Q_i\right] = q_o(t_{i+1}, u_i) \cdot \alpha(t_{i+1})$$

et l'horizon de la commande u_i est constitué par l'intervalle (t_i, t_{i+1}).

Par conséquent la loi de commande $\mathcal{L}_N^*$ est sous-optimale pour min C2, car il existe une suite $\mathcal{L}_N^o = \left\{u_i^o\right\}_o^N$ telle que

$$(11) \qquad E\left[\sum_{i=o}^{N} L(u_i)/Q_i\right] \leqslant E\left[\sum_{i=o}^{N} L(u_i^*)/Q_i\right]$$

Théorème 6:

La loi de commande $\mathcal{L}_N^*$ du taux d'infaisabilité minimum est meilleure que toute loi $\mathcal{L}_N$ acceptable, différente de $\mathcal{L}_N^o$, pour la minimisation du critère C2.

Pour démontrer l'inégalité (8) que nous écrivons

$$(8') \qquad E\left[\sum_{i=o}^{N} L(u_i)/Q_o\right] \leqslant E\left[\sum_{i=o}^{N} L(u_i^*)/Q_o\right]$$

nous allons d'abord démontrer le Théorème suivant, plus général:

Soit $\left\{X_n,\ n=0,\ldots.N\right\}$ un processus markovien et soit $U_n = \left\{u_n\right\}$ l'ensemble des commandes acceptables à la date t_n pour la minimisation du critère $E\left[\sum_{i=n}^{N} L(u_i) \,/\, x_n\right]$, $t_i \geqslant t_n$, où $E\left[L(u_i) \,/\, x_n\right]$ est l'espérance du coût de la transition de X_n à X_{i+1} en utilisant la commande u_i, connaissant $X_n = x_n$.

Soit u_i^* la commande appartenant à U_i telle que pour tout i vérifiant $t_i \geqslant t_n$ l'on ait

$$(12) \qquad E\left[L(u_i)\ /x_n\right] \geqslant E\left[L(u_i^*)\ /\ x_n\right]$$

Théorème 7:

La loi de commande définie par la séquence des commandes $\left\{u_i^*/x_n\right\}_{i=n}^{N}$ est meilleure que celle définie par $\left\{u_i\ /\ x_n\right\}_{i=n}^{N}$ pour la minimisation du critère $E\left[\sum_{i=0}^{N} L(u_i)/x_o\right]$ c'est à dire

$$(8'') \qquad E\sum_{i=0}^{N} L(u_i)/x_o \qquad \geqslant E\sum_{i=0}^{N} L(u_i^*)/x_o$$

Pour la démonstration de ce théorème nous utiliserons la propriété liée à la diffusion du processus markovien $\left\{X_n,\ n=0,\ldots,N\right\}$ donnée par

$$(13) \qquad E\left[L(u_i)\ /\ x_k\right] \geqslant E\left[L(u_i)/x_j\right]$$

pour i, k et j tels que $t_k \leqslant t_j \leqslant t_i$.

Ecrivons:

$$(14) \qquad E\left[\sum_{i=n+1}^{N} L(u_i)/x_{n+1}\right] = E\left[L(u_{n+1})/x_{n+1} + \ldots\ldots\ldots + L(u_{N-1})/x_{n+1}\right]$$

expression qui d'après (13) est majorée par

$$(15) \qquad E\left[\sum_{i=n+1}^{N} L(u_i)/x_n\right] = E\left[L(u_{n+1})/x_n + \ldots\ldots\ldots + L(u_{N-1})/x_n\right]$$

En portant terme à terme sur (15) l'inégalité (12) il vient:

$$(16) \qquad E\left[\sum_{i=n+1}^{N} L(u_i)/x_n\right] \geqslant E\left[\sum_{i=o}^{N} L(u_i^*)/x_n\right]$$

et en ajoutant des deux côtés de l'expression (16) ci-dessus le terme $E\ L(u_n^*)/x_n \geqslant 0$, il vient

$$(17) \qquad E\left[L(u_n)/x_n\right] + E\left[\sum_{i=n+1}^{N} L(u_i)/x_n\right] \geqslant E\left[\sum_{i=n}^{N} L(u_i^*)/x_n\right]$$

Mais d'après (12), le membre à gauche de l'expression(17) ci-dessus est majoré par

$$(18) \qquad E\left[L(u_n)/x_n\right] + E\left[\sum_{i=n+1}^{N} L(u_i)/x_n\right] = E\left[\sum_{i=n}^{N} L(u_i)/x_n\right]$$

de sorte que nous obtenons finalement

$$(19) \qquad E\left[\sum_{i=n}^{N} L(u_i)/x_n\right] \geqslant E\left[\sum_{i=n}^{N} L(u_i^*)/x_n\right]$$

Supposons maintenant que l'expression (19) ci-dessus soit vraie pour t_{n+1}. Comme il vient d'être démontré qu'elle est vraie pour t_n, par récurrence en arrière sur n, on montre que (19) est vraie pour t_o, c'est à dire que

$$(20) \qquad E\left[\sum_{i=o}^{N} L(u_i)/x_o\right] \geqslant E\left[\sum_{i=o}^{N} L(u_i^*)/x_o\right]$$

ce qui démontre le théorème 7

Pour la démonstration du théorème6il suffit de remarquer que la commande du taux d'infaisabilité minimum satisfait bien à l'expression (12) puisque nous avons:

$$(21) \qquad E\left[L(u_n)/\ Q_n\right] \geqslant E\left[L(u_n^*)/\ Q_n\right]$$

et que le processus d'accumulation $\{Q_n\}$ est markovien comme il a été vu au chapitre 3, ce qui permet d'utiliser la propriété en (13) et d'écrire finalement:

$$(22) \qquad \sum_{i=o}^{N} q_o(u_i) \geqslant \sum_{i=o}^{N} q_o(u_i^*)$$

La sous-optimalité des lois de commande à horizon (t_n, t_{n+1}) est en particulier intéressante pour les processus à horizon fuyant puisqu'il suffit à chaque date t_n de choisir la commande u_n^* qui minimise $E\left[L(u_n)/u_n\right]$.

Dans le cas du processus d'accumulation $Q(t)$ considéré ici la simplicité de calcul du taux d'infaisabilité $q_o(t)$ se prête bien à l'utilisation d'une loi de commande du type $\mathcal{L}_N^*$, qui comme il vient d'être démontré améliore la performance d'une loi de commande du type "test d'acceptation" où la dernière demande entraînant l'infaisabilité de la file accrue est rejetée.

4.3.- Réalisation de la solution sous-optimale

Ce paragraphe a pour but de synthétiser les résultats obtenus jusqu'ici en vue de définir les séquences de base d'un algorithme de commande qui satisfasse aux critères C1, C2 et C3.

Sachant que la discipline d'orientation σ (ou SDFS) est optimale pour le critère C1 de respect des délais d'achèvement individuels, celle-ci est utilisée pour déterminer la position $i' = \sigma(J_n, Q(t_n))$ de chacune des demandes J_n arrivant à l'UC aux dates $t=t_n$, $n=0, 1, \ldots,$.

Le critère C3 intervient lorsque la file d'attente orientée $Q'(t)$ devient infaisable après insertion d'une nouvelle demande J_n.

En rejetant n'importe quel élément de l'ensemble $V(t)$ des éléments $[j]$ pouvant être ignorés, $V(t)$ étant construit à partir de l'ensemble $D(t)$ des éléments inacceptables de la file accrue $Q'(t)$, l'on définit une commande u_n^j pas à pas qui satisfait le critère 3.

L'ensemble U_n tel que $u_n^j \in U_n$ est l'ensemble des commandes acceptables pour C2.

A chacune de ces commandes u_n^j correspond une valeur du taux d'infaisabilité $q_o(u_n^j)$ qu'il est possible de calculer au moyen des intervalles d'infaisabilité.

L'élément $[j^o]$ de $V(t)$ pour lequel $q_o(u_n^j)$ est minimum définit une commande aoptimale à la date t_n.

Connaissant $Q(t)$, l'expression de l'ensemble d'infaisabilité à la date $t+dt$, est donnée par les relations suivantes:

$$(14)\quad \begin{cases} I_{i',1}(t+dt) = RT_{i+j-1}(t),\ RT_{i+j-1}(t) - dt \\ I_{i',2}(t+dt) = FT^{m}_{i+j-1}(t), \\ I_{i',3}(t+dt) = 0,\ SP_{i+j-1}(t) - dt \end{cases}$$

où j est la position la plus proche de la tête de file Q(t) telle que $0 < dt < SP_j(t)$

Les variables aléatoires FT et PT sont supposées être indépendantes et identiquement distribuées et comme PT+ FT = RT, la fonction de probabilité G de RT permet de calculer $q_o(t+dt)$ à l'aide des expressions (14).

Pour la recherche de la commande u_n^j qui minimise $q_o(t+dt)$, le calcul est répété pour les p(t) éléments de V(t), c'est à dire pour chacune des p(t) commandes de U_n.

La demande pour laquelle $q(u_n^j)$ est minimum est rejetée de la file $Q'(t_n)$ qui devient alors $Q'(u_n^j)$.

Dans le cas où il existe plus d'une commande u_n^j minimisant $q_o(u_n^j)$, il faudrait alors définir un critère supplémentaire de choix.

Ce cas n'est pas traité ici et n'importe laquelle de ces commandes est utilisée.

4.4.- Etude d'un cas particulier important

Dans tout ce qui précède et de façon à donner à cette étude une portée suffisament générale, les variables du processus d'accumulation concernant les composantes aléatoires de chaque demande $J_n = (PT, FT)$ ont été prises indépendantes.

Il existe cependant un cas important où PT et FT sont liés par la relation linéaire.

(15) $\quad RT = K \cdot PT$

où K est une constante appelée coefficient d'urgence et telle que $K > 1$. L'expression en (15) alloue un temps de réponse proportionnel au temps de service (Voir [7] p. 97) et permet d'autre part de définir RT d'une manière raisonnable lorsque PT uniquement est bien connu.

La constante K est la même pour toute les demandes appartenant à T.

La relation de proportionalité (15) n'est pas en fait restrictive puisque dans le cas général où PT et FT sont mutuellement indépendantes, l'urgence moyenne représentée par la moyenne de la distribution calculée de RT est en fait l'équivalent du coefficient d'urgence K dont la valeur est commune à tous les éléments J_n de T.

Dans les deux cas il est alors possible de définir des classes de demandes d'après le coefficient d'urgence fixé pour chaque classe, c'est à dire la moyenne de la distribution de RT dans le cas général et la valeur de K dans le cas considéré ici.

Etant donné la relation RT = FT + PT, l'expression du temps de réponse RT dans (15) devient

(16) $\quad FT = (K - 1) \cdot PT$

de sorte que chacune des demandes $J_n \in T$ est complètement définie par la donnée de PT uniquement et $J_n = J_n(PT)$.

Avec les relations (15) et (16), la nième demande J_n arrivant à la date $t=t_n$ entraîne l'infaisabilité de la file accrue s'il existe une position i dans Q(t) telle que

$$(17) \quad \left\{ RT_i(t) < K.PT \leqslant RT_{i\ 1}(t) \text{ et} \left(PT > FT_i^m(t) \text{ et/ou } (K-1).PT \leqslant SP_i(t)\right)\right\}$$

et les expressions des intervalles d'infaisabilité $I_{i',1}(t)$, $I_{i',2}(t)$ et $I_{i',3}(t)$ de la file d'attente accrue sont:

Pour i' = 2, ... N'(t) - 1. (Voir 3.2.1)

$$(18) \quad \begin{cases} I_{i',1}(t) = \left]\dfrac{RT_i(t)}{K} \ , \ \dfrac{RT_{i\ 1}(t)}{K}\right] \\ I_{i',2}(t) = \left]FT_1^m\ (t) \ , \ \infty\right[\\ I_{i',3}(t) = \left]0 \ , \ \dfrac{SP_i(t)}{K-1}\ \right] \end{cases}$$

Pour i' = 1

$$(19) \quad \begin{cases} I_{11}(t) = \ \left]\ 0 \ , \ \dfrac{RT_1(t)}{K}\ \right] \\ I_{12}(t) = \ \left]FT_1^m\ (t) \ , \ \infty\right[\\ I_{13}(t) = \ ]0, \ 0[\end{cases}$$

et pour i' = N'(t)

$$(20) \quad \begin{cases} I_{N1}(t) = \left]\dfrac{RT_N(t)}{K} \quad , \ \infty\right[\\ I_{N2}(t) = \ ]\infty \ , \ \infty[\\ I_{N3}(t) = \left]0 \ , \ \dfrac{SP_N(t)}{K-1}\right[\end{cases}$$

Les intervalles ci-dessus portent sur la seule variable PT de sorte que l'on cherche la position i' = i:

$$(21) \qquad \left\{ PT \in I_{i',1}(t) \text{ et } PT \in I_{i',2}(t) \text{ et/ou } PT \in I_{i',3}(t) \right\}$$

ou encore

$$(21') \qquad \left\{ PT \in I_{i',1}(t) \cap (I_{i',2}(t) \cup I_{i',3}(t)) \right\}$$

L'ensemble d'infaisabilité $I_{i'}(t)$ correspondant à une position i' de Q'(t) porte sur la seule variable PT et peut s'écrire

$$(22) \qquad I_{i'}(t) = I_{i',p}(t) \cap I_{i',F}(t) \qquad \text{avec}$$

$$I_{i',p}(t) = I_{i',1}(t) \qquad \text{et} \qquad I_{i',f}(t) = I_{i',2}(t) \cup I_{i',3}(t)$$

Or par définition, les intervalles $I_{i',p}(t) = I_{i',1}(t)$ sont des intervalles disjoints et il en est de même de $I_{i'}(t)$ i'=1, ..., N'(t).

Par conséquent, pour l'évènement $\left\{ J_n \in I_1(t) \cup J_n \in I_2(t) \ldots \cup J_n \in I_{N'}(t) \right\}$, il est possible d'écrire :

$$(23) \qquad \left\{ J_n \in I_1(t) \cup \ldots \quad J_n \in I_{N'}(t) \right\} \equiv \left\{ J_n \in (I_1(t) \cup \ldots \cup I_{N'}(t) \right\}$$

et en définissant l'ensemble d'infaisabilité pour Q(t) par

$$(24) \qquad I(t) = \bigcup_{i=1}^{N'(t)} I_i(t)$$

l'expression (25) devient, avec $J_n = J_n(PT)$

$$(25) \qquad \left\{ J_n \in \bigcup_{i=1}^{N'(t)} I_i(t) \right\} = \left\{ J_n \in I(t) \right\} = \left\{ PT \in I(t) \right\}$$

qui n'est plus, comme dans le cas général, une notation symbolique mais une égalité mathématique.

L'expression du taux d'infaisabilité $q_o(t)$ est déterminée par la connaissance des fonctions de probabilité des variables aléatoires Δ et PT.

Comme Δ appartient à une distribution exponentielle négative de paramètre λ (Voir 3.1.) la probabilité $q_o(t)$ devient:

$$(26) \qquad q_o(t) = \lambda \cdot P_r \left[PT \in I(t+dt) \;/\; Q(t) \right]$$

Ainsi, avec la relation de proportionalité entre RT et PT, l'expression du taux d'infaisabilité est simple à calculer.

D'une façon analogue à celle de 3.6, il est possible alors de rejeter de $Q'(t)$, l'élément j^o de $V(t)$ qui fournit la valeur minimum de $q_o(u_n^j)$.

En conclusion, dans ce chapitre 4 nous avons montré que la loi de commande du taux d'infaisabilité minimum est meilleure que toute autre loi de commande acceptable pour la minimisation du critère C2, sauf bien sûr la loi de commande optimale.

Cette loi sous-optimale est constituée par la séquence des commandes qui à chaque transition de $Z(t)$ à l'état 0 minimisent le taux d'infaisabilité.

CHAPITRE 5

5. La simulation RTSIM

L'étude faite dans les chapitres précédents a abouti à la réalisation d'un modèle mathématique simple du système constitué par l'UC et la séquence E d'arrivées des demandes de traitement.

A l'aide de ce modèle il a été possible de determiner d'une part la loi d'ordonnancement optimale pour le critère C1 de faisabilité et d'autre part, dans le cas dynamique, de montrer que la loi de commande $_N$ du taux d'infaisabilité minimum est meilleure que la loi de commande $_N$ définie par le test d'acceptation, pour la minimisation du critère $C2 = E(C(0,t))$.

Ainsi posé et sous les hypothèses établies en début de mémoire, le problème théorique a été résolu.

L'étape suivante qui fait l'objet de ce chapitre concerne l'étude expérimentale des solutions proposées.

Cette étude expérimentale est basée sur une simulation réalisée à partir du modèle mathématique obtenu et est constituée par un programme en langage FORTRAN IV appelé RTSIM.

Le but de RTSIM est principalement de tester la sous-optimalité de la loi de commande du taux d'infaisabilité minimum défini en 3.4 mais aussi de simuler les processus d'accumulation du type défini en 3.1 et 3.2.

Cependant, RTSIM a été écrit pour le cas où les durées d'achèvement RT sont proportionnelles aux temps de service comme il a été vu en 4.4 du chapitre 4.

Les résultats discutés ici correspondent donc à cette situation.

5.1.- Description de RTSIM

La simulation RTSIM est constituée d'un ensemble de trois programmes écrits en langage FORTRAN IV dont il existe deux versions. Une première pour le calculateur CDC6600 et une deuxième pour le calculateur IBM 1130 et autres machines compatibles.

RTSIM est en fait le nom du programme principal de la simulation qui utilise deux sous programmes: ALGHJ et ALGDJ.

D'une manière générale RTSIM simule le processus d'accumulation désiré alors que ALGHJ et ALGDJ réalisent l'action de commande correspondant à la méthode du test d'acceptation et à celle du taux d'infaisabilité minimum respectivement.

Les fonctions réalisées par RTSIM sont les suivantes:

a) Génération des demandes aléatoires (temps inter-arrivées et temps de service PT) de la séquence d'entrée E.

b) Choix de la discipline D d'ordonnancement

c) Choix de la loi de commande (ALGHJ, ALGDJ)

d) Décrémentation du temps de service de l'élément en tête de file et passage à l'exécution de la demande suivante, entre l'arrivée de demandes successives. Si la file d'attente est vide, exécution d'une boucle d'attente.

e) Insertion d'une nouvelle demande dans la file d'attente lorsque sa date d'arrivée est égale ou inférieure à l'heure courante.

f) Test de faisabilité de la file d'attente accrue.

g) En cas d'infaisabilité, appel de sous programme de commande choisi et incrémentation d'une unité du nombre de demandes insatisfaites.

h) En cas de faisabilité de la file accrue, si le temps de simulation n'est pas complété, retour à d), sinon

i) Impression des résultats de la simulation

Les fonctions réalisées par le sous programme ALGHJ sont les suivantes:

j) Refus de la demande courant (Voir g) ci-dessus)

k) Retour à RTSIM en d) ci-dessus.

Les fonctions réalisées par le sous programme ALGDJ sont les suivantes:

l) Recherche de tous les éléments inacceptables de la file d'attente accrue.

m) Recherche des p(t) éléments de l'ensemble U(t) des commandes acceptables.

n) Test du rejet d'un élément de V(t).

o) Calcul des intervalles d'infaisabilité correspondants, à la date t + dt.

p) Calcul du taux d'infaisabilité correspondant.

q) Passage à l'élément suivant de V(t), jusqu'au dernier et retour à n)

r) Recherche du taux d'infaisabilité minimum $q(u^j(t))$

s) Rejet de l'élément correspondant de V(t)

t) Réarrangement de la file d'attente.

u) Retour à RTSIM en d) ci-dessus.

La génération des demandes aléatoires considérée en a) est basée sur les techniques classiques pour la génération pseudo-aléatoire sur calculateur numérique de réels y identiquement distribués et tels que

(1) $0. \leqslant y \leqslant 1.$

Pour la génération des y entre 0. et 1., RTSIM utilise la fonction RANF de la bibliothèque du calculateur CDC 6600 ou le sous programme RANDU de la bibliothèque du calculatuer IBM 1130, suivant la version utilisée.

A partir de RANF ou de RANDU il est alors possible de générer des séquences de réalisation de variables aléatoires appartenant à des fonctions de probabilité données.

Les fonctions de probabilité $F_1(\lambda, t)$ et $F_2(\mu, s)$ choisies pour la génération des variables aléatoires Δ et PT introduites en 3.1 sont exponentielles négatives:

(2) $$F_1(\lambda, n) = Pr[\Delta \leqslant r] = 1 - e^{-\lambda t} \qquad t \geqslant 0$$

et

(3) $$F_2(\mu, s) = Pr[PT \leqslant s] = 1 - e^{-\mu s} \qquad s \geqslant 0$$

Les expressions (2) et (3) constituent des conditions suffisantes d'indépendance de la séquence $\{F(t),\ t \geqslant 0\}$ du processus de faisabilité introduit en 3.2.

La génération d'une valeur t ou d'une valeur s appartenant respectivement à (2) et (3) consiste à trouver la solution de l'équation:

(4) $y = 1 - e^{-\lambda t}$

où y est un nombre réel satisfaisant à (1).

La valeur de y est donnée par la fonction RANF, ce qui fournit pour la réalisation t de

(4') $t = -\text{Log}_e(1 - y) / \lambda$

Naturellement une telle méthode implique l'existence d'une bijection entre l'ensemble des y et celui des t, ce qui est le cas pour les distributions exponentielles négatives telles que (2) et (3).

D'après la relation linéaire (50) entre PT et RT, la génération des temps de service PT fixe les valeurs associées des temps de réponse RT.

En ce qui concerne la discipline d'ordonnancement, RTSIM peut simuler des processus d'accumulation basés sur les disciplines suivantes:

-Orientation (SDFS)
-Plus petit d'abord (SJFS)
-Premier arrivé, Premier servi (FCFS)

Il suffit pour cela d'indiquer le code correspondant en début de simulation au moyen de la variable IDISCPL (Voir A1)

Le choix de la loi de commande est aussi fixé en début de simulation par la variable ICNTROL (Voir A1)

Le mode d'emploi du programme RTSIM est donné dans l'annexe A1, en fin du mémoire.

5.2.- Comparaison des performances de ALGDJ et ALGHJ

Le but principal des résultats obtenus à l'aide de RTSIM est

de pouvoir comparer les performances de l'algorithme de commande du taux d'infaisabilité minimum (ALGDJ) avec l'algorithme de rejet de M.M. Johnson (ALGHJ).

La comparaison des performances de ces deux algorithmes porte sur le nombre total C(0,t) de demandes rejetées à la fin de chaque simulation d'une durée (0,t).

Ce nombre constitue la mesure de performance utilisée.

RTSIM peut générer différents processus d'accumulation et différentes séquences d'arrivée E. Pour une simulation donnée (discipline et séquence E fixées) RTSIM est exécuté deux fois, une pour chacun des algorithmes ALGDJ et ALGHJ.

Telle qu'elle a été utilisée par l'auteur, la simulation est répétée pour les NAK valeurs du coefficient d'urgence AK liant RT et PT et pour quatre valeurs décroissantes de la moyenne μ de la fonction de répartition $F(\mu,\sigma)$ des temps de service PT.

Toutes ces combinaisons sont décrites par les cartes données de RTSIM (Voir annexe A1)

Les valeurs finales de la mesure C(0,t) obtenues pour quatre valeurs différentes du coefficient $\rho = \lambda/\mu$ sont celles des tables 5, 6, 7 et 8, où la moyenne λ de la suite $\{\Delta\}$ des temps entre arrivées est maintenue fixe et égale à 14,3 et où les quatre valeurs de la moyenne sont respectivement μ = 7, 14.1, 14.9, 22.2.

Pour chaque simulation présentée ici, la durée est de IEND = 1000 unités DT = 0.1 seconde; la séquence d'arrivée E est constituée de 64 demandes J_n.

Le coefficient d'urgence AK prend successivement les NAK =15 valeurs 1.5, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15.

Pour la valeur AK = 1.5 du coefficient d'urgence (grande urgence) les algorithmes ALGDJ et ALGHJ fournissent à peu près le même nombre C(O,t) de demandes rejetées, pour des valeurs croissantes du coefficient de trafic RO.

Ceci s'explique par le fait que pour RO 1 le processus d'accumulation présente un petit nombre N(t) d'éléments en moyenne et lorsqu'une transition à l'état $\mathcal{Z}(t) = 0$ se produit, le nombre p(t) d'éléments de V(t) à tester est petit.

Pour RO < 1, comme AK = 1.5 est petit, le nombre de transitions à l'état $\mathcal{Z}(t) = 0$ est relativement élevé, ce qui entraîne que le nombre d'éléments N(t) du processus d'accumulation n'est jamais très élevé en moyenne, puisque ALGDJ et ALGHJ rejettent alors une demande pour chaque transition à $\mathcal{Z}(t) = 0$.

Par conséquent le nombre p(t) d'éléments de V(t) est petit aussi.

Ainsi donc pour AK = 1.5 (RO < 1 et RO $\geqslant 1$) ALGDJ apporte peu d'améliorations au rejet de la demande J_n qui cause l'infaisabilité de la file d'attente accrue, tel qu'il est réalisé par ALGHJ.

Pour des valeurs plus grandes de AK et pour RO petit, la situation considérée plus haut se reproduit et ALGDJ n'améliore pas la performance de ALGHJ, car le nombre p(t) d'éléments de V(t) est petit avec N(t).

Par contre, pour les valeurs croissantes du coefficient de trafic RO, le nombre N(t) d'éléments du processus d'accumulation croît avec le nombre p(t) d'éléments de V(t).

Bien entendu, pour RO donné, le nombre C(O,t) d'éléments rejetés diminue avec 1/AK, c'est à dire avec l'urgence des tâches.

DISCIPLINE D'ORIENTATION SDFS

RO = 0.490

AK	1.5	2	3	4	5	6	7	8	9	10	11	12	13	14	15
ALGDJ	7	3	0												
ALGHJ	7	3	0												

Table 1

RO = 0.986

AK	1.5	2	3	4	5	6	7	8	9	10	11	12	13	14	15
ALGDJ	9	8	2	0											
ALGHJ	10	14	4	0											

Table 2

RO = 1.045

AK	1.5	2	3	4	5	6	7	8	9	10	11	12	13	14	15
ALGDJ	10	9	2	0											
ALGHJ	10	10	6	0											

Table 3

RO = 1.556

AK	1.5	2	3	4	5	6	7	8	9	10	11	12	13	14	15
ALGDJ	12	10	6	3	1	0									
ALGHJ	14	19	12	8	7	0									

Table 4

Les résultats obtenus au cours de cette simulation reflètent l'amélioration que d'une façon globale ALGDJ apporte par rapport à la performance de ALGHJ en particulier pour des traffics élevés de la séquence d'arrivée E.

D'autres résultats intéressants fournis par RTSIM concernent l'utilisation d'autres disciplines d'ordonnencement, en particulier FCFS (ou premier arrivé premier servi) et SJFS (ou le plus petit d'abord), pour les même séquences d'arrivée E que pour la discipline d' Orientation SDFS.

Tout d'abord, en comparant les valeurs de C(O,t) fournies par l'utilisation des disciplines SDFS (Tables 1, 2, 3, 4) et FCFS (tables 5, 6, 7, 8) avec l'algorithme ALGHJ, c'est à dire en refusant toute demande qui cause l'infaisabilité de la file accrue, il apparaît clairement que la discipline FCFS est loin de minimiser le nombre total de demandes rejetées C(O,t).

En ce qui concerne les performances comparées de ALGDJ et ALGHJ pour la discipline FCFS, les explications des résultats obtenus sont analogues à celles pour la discipline SDFS.

Tant que le nombre d'éléments N(t) de la file d'attente reste petit -soit parce que RO est petit, soit parce que AK est petit pour RO correspondant à des traffics importants- ALGDJ n'améliore pratiquement pas la performance de ALGHJ.

Par contre, dès que N(t) devient plus grand avec le nombre d'éléments p(t) de V(t), ce qui correspond à des valeurs élevées du coefficient d'urgence AK (faible urgence) pour RO $>$ 1

DISCIPLINE FCFS

RO = 0.490

AK	1.5	2	3	4	5	6	7	8	9	10	11	12	13	14	15
ALGDJ	15	13	11	10	9	9	9	8	6	5	5	5	5	5	4
ALGHJ	15	13	11	10	9	9	9	9	7	5	5	5	5	5	4

Table 5

RO = 0. 986

AK	1.5	2	3	4	5	6	7	8	9	10	11	12	13	14	15
ALGDJ	24	27	23	24	24	21	16	16	17	11	11	10	9	8	8
ALGHJ	24	27	26	25	24	21	19	19	17	17	17	15	14	11	10

Table 6

RO = 1.045

AK	1.5	2	3	4	5	6	7	8	9	10	11	12	13	14	15
ALGDJ	24	25	26	25	23	21	18	16	16	12	12	10	9	9	9
ALGHJ	25	28	28	27	26	24	21	19	19	19	17	16	16	15	15

Table 7

RO = 1.556

AK	1.5	2	3	4	5	6	7	8	9	10	11	12	13	14	15
ALGDJ	31	36	31	27	23	21	18	17	16	16	15	14	12	11	11
ALGHJ	32	38	39	36	33	32	32	32	29	28	28	26	26	26	24

Table 8

alors ALGDJ améliore nettement les performances fournies par ALGHJ.

Pour une valeur fixée du coefficient de traffic RO, C(O,t) diminue avec 1/AK, mais bien plus lentement qu'avec la discipline d'orientation SDFS.

Pour la discipline SJFS du plus petit d'abord, les résultats obtenus pour les quatre valeurs de RO habituelles sont données par les tables 9, 10, 11, 12.

La première constatation est que les valeurs du nombre C(O,t) de valeurs rejetées obtenues avec la discipline SJFS sont exactement les mêmes que celles obtenues avec la discipline SDFS (pour RO=0.490) ou presque les mêmes (RO=0.986 et RO=1.045).

Ceci s'explique par le fait que le nombre d'éléments N(t) de la file d'attente pour les valeurs considérées de RO est petit et comme RT = K. PT, les files d'attentes $Q_{SDFS}(t)$ et $Q_{SJFS}(t)$ sont à peu près les mêmes.

Pour RO = 1.556, C(O,t) prend de nouveau à peu près les mêmes valeurs qu'avec la discipline d'orientation, mais il faut arriver à des valeurs plus grandes du coefficient d'urgence AK (AK = 10 pour la discipline SJFS contre AK = 6 pour la discipline SDFS) pour obtenir des files d'attentes faisables sur tout l'intervalle (O,t).

DISCIPLINE SJFS

RO = 0.490

AK	1.5	2	3	4	5	6	7	8	9	10	11	12	13	14	15
ALGDJ	7	3	0												
ALGHJ	7	3	0												

Table 9

RO = 0.986

AK	1.5	2	3	4	5	6	7	8	9	10	11	12	13	14	15
ALGDJ	10	8	2	1	0										
ALGHJ	11	14	2	1	0										

Table 10

RO = 1.045

AK	1.5	2	3	4	5	6	7	8	9	10	11	12	13	14	15
ALGDJ	11	9	2	1	0										
ALGHJ	10	10	2	1	0										

Table 11

RO = 1.556

AK	1.5	2	3	4	5	6	7	8	9	10	11	12	13	14	15
ALGDJ	13	10	7	3	1	2	1	1	1	0					
ALGHJ	14	19	11	7	1	8	8	1	1	0					

Table 12

En conclusion, les résultats obtenus par la simulation RTSIM permettent de confirmer expérimentalement les points suivants:

Premièrement la discipline d'orientation SDFS est "meilleure" que les disciplines SJFS et FCFS en ce sens que les valeurs de C(0,t) obtenues pour les trois disciplines à l'aide de l'algorithme ALGHJ basé sur le test d'acceptation (Voir 2.5) sont telles que

$$(6) \qquad C(0,t)_{SDFS} \leqslant \min \left(C(0,t)_{SJFS} , C(0,t)_{FCFS}\right)$$

avec les notations habituelles. Or l'on sait que le nombre C(0,t) obtenu avec l'algorithme ALGHJ est égal au nombre de demandes de la séquence E qui causent l'infaisabilité du processus d'accumulation.

Donc à chaque arrivée d'une telle demande, la discipline SDFS satisfait bien le Théorème 4 et à l'expression (24) du chapitre 2 d'optimalité pour le critère C1.

Deuxièmement, la loi de commande du taux d'infaisabilité minimum, simulée par ALGDJ est meilleure que la loi d'acceptation simulée par ALGHJ qui est une commande acceptable pour min C2 en accord avec les résultats théoriques fournis au chapitre 4.

CHAPITRE 6

6.- Conclusions

Dans les chapitres précédents il a été montré comment il est possible, en utilisant un modèle mathématique simple, de résoudre un des problèmes posés par l'ordonnancement dynamique des demandes de traitement sur un monoprocesseur, lorsque celles-ci sont soumises à des contraintes de délai d'achèvement critique.

Il a été montré au chapitre 2 que la discipline d'orientation SDFS est optimale pour le critère C1 de respect des délais d'achèvement individuels des éléments d'une file d'attente statique donnée.

Cependant, une file statique orientée peut devenir infaisable ce qui implique la perte des éléments inacceptables qui atteignent la tête de file, d'où la definition de L_n.

Au chapitre 3, l'introduction de données aléatoires pour la définition de la séquence d'arrivées E a permis l'étude dynamique du processus d'accumulation $Q(t)$ et des conditions sous lesquelles le critère de faisabilité C3 peut être satisfait.

Ensuite, le processus de faisabilité $\{Z(t),\ t \geqslant 0\}$ associé au processus d'accumulation $\{Q(t),\ t \geqslant 0\}$ a été défini comme l'observation des états faisables/infaisables de $Q(t)$.

Au chapitre 4 il a été montré qu'à chaque date t d'arrivée d'une nouvelle demande J_n telle que $Z(t_n) = 0$, il existe un ensemble U_n de commandes acceptables u_n^j applicables à l'instant t_n qui satisfont le critère C2.

L'ensemble des séquences de ces commandes constitue l'ensemble $\mathcal{L}$ des lois de commandes acceptables pour la minimisation du critère $C2 = E\left[C(0,t)\right]$ où la variable aléatoire $C(0,t)$ représente le nombre d'éléments ignorés par l'UC sur l'intervalle $(0,t)$ sous les hypothèses:

H.3 $m(t_n) = 1, \forall\ t_n$ t.q. $Z(t_n) = 0$

H.4 $Z(t) = 1, \forall\ t \neq t_n$, $n = 0,...$

Finalement, il a été montré qu'il existe une loi de commande sous optimale pour min. C2 mais qui améliore la performance de la loi de commande du test d'acceptation.

Cette loi est définie par la suite des commandes acceptables qui à chaque transition de $Z(t)$ à l'état 0, minimisent le taux d'infaisabilité $q(u_n^j)$.

Au chapitre 5, la solution proposée a été programmée en FORTRAN et utilisée par la simulation RTSIM. Ainsi les disciplines d'ordonnancement SDFS, FCFS et SJFS de même que la loi de commande du taux d'infaisabilité minimum ont pu être testées pour la satisfaction des critères C1 et C2.

Cependant, les résultats obtenus au cours de cette étude sont basés sur un certain nombre d'hypothèses qu'il convient de discuter.

a) La loi de commande sous optimale présentée ici repose sur l'hypothèse que tous les éléments J_n qui constituent la séquence d'arrivées E ont la même valeur pour UC, en ce sens que le rejet définitif d'une demande cause la même déterioration du service fourni à l'environnement temps réel critique P et ceci quelles que soient les caractéristiques individuelles de la demande refusée.

Dans un grand nombre d'applications réelles cette hypothèse n'est pas acceptable, puisque par exemple, des demandes de traitement correspondant à des alarmes de sécurité ne peuvent pas être rejetées sans tenir en compte leur nature.

Une première solution consiste à diviser l'ensemble des demandes en deux classes. L'appartenance de toute demande à une classe constitue un nouveau paramètre qui doit figurer de façon explicite dans

l'expression J_n définie en 2.2.

Les demandes j de la classe supérieure où privilégiée faisant partie de l'ensemble V(t) sont telles que les commandes $u_n^j \in U_n$ correspondantes ne sont pas considérées pour le calcul du taux d'infaisabilité minimum et sont par conséquent exclues de l'ensemble U_n des commandes acceptables.

Ainsi les demandes de la classe privilégiée ne peuvent pas être rejetées d'une file d'attente devenue infaisable.

De façon à prevoir le cas où seules des demandes de la classe privilégiée appartiendrait á V(t) pour certaines transitions de Z(t) à l'état 0, la possibilité de rejeter $m(t) \geqslant 1$ demandes de la classe commune ou non privilégiée pourrait être envisagée.

Il est à noter que la partition de l'ensemble T considérée ci-dessus revient à définir deux classes de priorités -priorités d'achèvement- différentes. Cependant le paramètre d'appartenance à la classe privilégiée ou prioritaire n'est utilisé que dans les cas où la demande correspondante appartient à l'ensemble V(t).

Ceci est différent du mécanisme de priorités -priorités d'accès- généralement utilisé, puisque ce dernier est utilisé en fait pour l'ordonnancement des demandes.

D'après l'optimalité de la discipline d'orientation pour le critère de faisabilité C1, tout ordonnancement basé sur un système de priorités fixes n'est pas optimal pour C1.

D'une manière plus générale, la partition de l'ensemble T des demandes pourrait être étendue à M classes différentes i, toujours au sens d'une commande de rejet.

Le critère C2 á minimiser pourrait alors être une fonction des espérances E $C_i(0,t)$ i=1, ...M.

b) La séquence d'arrivées E a été munie d'une structure probabiliste de façon à être aussi générale que possible du point de vue de l'UC.

Il faut remarquer que cette généralité n'inclut pas en fait le cas des demandes périodiques. En effet dans ce cas le paramètre représentatif de la période d'apparition des demandes devrait aussi être explicité dans l'expression de J_n.

Les recherches entreprises par C.L. Liu et J.W. Layland[14] et J. Labetoulle [17] étudient cette situation et montrent que la discipline SDFS n'est plus optimale pour le critère de faisabilité C1.

c) Dans l'étude faite ici, il a été supposé que le temps d'exécution de l'algorithme de recherche de la commande u_n^j à chaque transition de Z(t) à l'état 0 est négligeable ce qui est inexact d'un point de vue pratique.

Dans la réalité il peut être admis que la durée d'exécution de l'algorithme ALGDJ est proportionnelle au nombre N(t) d'éléments dans la file d'attente accrue et au nombre p(t) d'éléments de l'ensemble V(t), en admettant que le temps nécessaire pour tester un élément de V(t) pour une position donnée de Q'(t dt) soit sonctant.

Par conséquent il est possible d'évaluer le temps d'unité centrale nécessaire au calcul de la valeur minimale de q(u'(t)) à chaque transition du processus d'infaisabilité à l'état Z(t) = 0.

Le temps d'UC peut alors être inclus dans le temps de service PT de la demande J_n venant d'arriver et les calculs habituels avec cette nouvelle valeur de PT peuvent alors être entrepris.

Dans certains cas où le temps de calcul de l'algorithme d'optimisation est néanmoins prohibitif il faudrait alors peut-être considerer une règle de décision au sujet de l'exécution complète de ALGDJ à partir du moment où sa durée d'exécution a été évaluée.

Par exemple il serait possible de fixer un seuil défini par un pourcentage du plus petit temps libre positif dans la file faisable avant insertion de J_n.

Si le rapport du temps de calcul évalué sur un temps libre minimum est supérieur au pourcentage établi, alors l'algorithme de commande n'est pas exécuté et la dernière demande J_n d'arrivée est refusée suivant le test d'acceptation en 2.4.

Sinon, l'algorithme ALGDJ est exécuté normalement.

Par ailleurs il est intéressant de remarquer qu'avec les possibilités techniques des mémoires ROM programmable (PROM) où les temps de lecture sont pratiquement réduits de moitié, le temps d'exécution de programmes de service du type ALGDJ devient moins prohibitif.

d) L'algorithme de commande ALGDJ présenté dans ce mémoire est basé sur le calcul du taux d'infaisabilité $q_o(u_n^j)$ sous l'hypothèse que l'action de rejet est réalisée à la date $t=t_n$ d'arrivée de la demande J causant l'infaisabilité.

Plus précisement, ALGDJ insère J_n dans la file d'attente et tout autre élément appartenant à $V(t)$ -pour $p(t) > 1$- peut être également rejeté en accord avec la définition de l'ensemble $U(t)$ des commandes acceptables pour le critère C2.

Ainsi donc, après constitution à la date $t=t_n$ de l'ensemble U_n des commandes acceptables pour le critère C2 il est toujours possible de rejeter un élément de $V(t_n)$ à une date t telle que:

(1) $\quad t_n < t \leqslant t'$

où t' est au plus égal à la date de début de traitement du dernier élément de $V(t_n)$, et de satisfaire C3.

Dans ces conditions, le choix de la commande $u(t) \in U(t)$ où t vérifie (1) ne dépend plus uniquement de l'expression du taux d'infaisabilité $q_o(u^j(t))$ minimum mais aussi de l'instant $t = t^o$ vérifiant (1).

Enfin, d'un point de vue théorique, il serait intéressant de résoudre le problème de la commande optimale pour le critère C2 = min E C(0,t) même si la mise en oeuvre de l'algorithme correspondant aboutit à des temps d'exécution prohibitifs et donc contraires, en pratique, à la satisfaction du critère de faisabilité C1.

En conclusion, l'étude présentée dans ce mémoire a pour but, d'une part de mettre en relief l'importance du concept délai d'achèvement individuel dûment explicité et qui permette de gérer les demandes d'un environnement temps réel critique d'une manière individuelle par opposition aux ordonnancements dont la tendance statistique uniquement intéresse, et d'autre part de montrer comment il est possible d'apporter des améliorations sensibles d'une commande sous optimale dont le coût en temps d'exécution est moins prohibitif que celui d'une commande optimale.

ANNEXES

A1. Mode d'emploi de RTSIM

Le mode d'utilisation de la simulation RTSIM est très simple et consiste seulement à constituer correctement la séquence des cartes données.

Les paramètres utilisés par RTSIM pour la définition d'une simulation sont les suivants:

a) IWRITE, INTWRIT
b) DT, M, ALAMDA
c) IGENST
d) TETAS, TETAM, AMU
e) NAK, AK
f) IDISCPL
g) ICNTROL
h) IEND

a) IWRITE et INTWRIT sont deux paramètres qui fixent l'impression des résultats intermédiaires et peuvent chacun prendre la valeur 0 ou 1 dans le format I1.

INTWRIT = 1
Imprime (Voir Fig. 1)

1) L'identificateur J_n et la date d'arrivée d'une nouvelle demande et dans le cas où la file d'attente accrue est infaisable,

2) L'identificateur J_n et la date de rejet de la demande perdue déterminée par ALGDJ ou ALGHJ.

IWRITE = 1
Imprime (Voir Fig. 2):

3) L'identificateur (J_n) et la date de fin de traitement de l'élément en tête de file ayant été complété.

4) L'état Q'(t) de la file d'attente accrue à l'arrivée d'une nouvelle demande.

En cas d'infaisabilité de Q'(t), pour les p(t) éléments de V(t):

5) L'identificateur J_n de l'élément de V(t) testé pour son rejet.

6) Les intervalles d'infaisabilité correspondant au rejet de $J_n \in V(t)$.

7) La valeur du taux d'infaisabilité correspondant au rejet de $J_n \in V(t)$.

8) Le nouvel état de la file d'attente après le rejet de l'élément de V(t) qui minimise le taux d'infaisabilité.

IWRITE = 0 Supprime les impresions 3) et 8)

INTWRIT = 0 Supprime les impressions 1) et 2) ce qui implique IWRITE = 0 et imprime les résultats finaux.

9) Les résultats finaux de la simulation (Voir Fig. 3)

Ainsi donc les combinaisons possibles pour l'impression de résultats intermédiaires illustratifs de l'évolution d'un processus d'accumulation donné sont les suivants (Tab. 1)

Messages imprimés	IWRITE	INTWRIT
9	0	0
1, 2, 9	0	1
1,2,3,4,5,6,7,8,9	1	1

Table 1

La combinaison (0,0) permet de connaître les résultats finaux de la simulation tels qu'ils apparaissent dans la Fig. 3.

La combinaison (0,1) permet de suivre la séquence d'arrivée des demandes, la séquence de rejet des éléments perdus et de connaître les résultats finaux.

La combinaison (1,1) permet de suivre l'état de la file d'attente entre les arrivées, à l'arrivée de chaque nouvelle demande (file accrue faisable ou infaisable), de connaître les intervalles d'infaisabilité, le taux $q(u^j(t))$ et les résultats finaux.

La combinaison (1,0) est interdite.

b) DT, M, ALAMDA

Le paramètre DT fournit la période de résolution de l'horloge CLOCK de RTSIM dans le format F10.1.

Le paramètre M dans le format I3 définit le nombre maximum de demandes J_n dans le cas où ce nombre est connu de l'unité centrale simulée.

Le paramètre ALAMDA définit la moyenne λ de la fonction de probabilité $F_1(\lambda, t)$ de la variavle aléatoire Δ dans le format F10.3.

DT,M, ALAMDA sont lus sur la même carte de donnée (F10.1, I3, F10.3).

c) IGENST est un paramètre qui choisit la fonction de probabilité de la variable aléatoire PT des temps de service. En effet, RTSIM permet de générer des temps de service appartenant à une fonction exponentielle négative du type $F_1(\mu, r)$ mais aussi à une distribution normale $N(\mu, \sigma^2)$.

IGENST peut prendre la valeur 1 ou 2 en format I1.

d) TETAS, TETAM, AMU

Les paramètres TETAS et TETAM représentent l'écart type et la moyenne de la fonction de répartition de la variable aléatoire PT,

lorsque celle-ci est choisie comme étant normale $N(\mu, \sigma^2)$.

TETAS et TETAM sont lus sur la même carte de donnée dans le format 2F10.1.

Le paramètre AMU définit la moyenne μ de la fonction de probabilité $F_2(\mu, r)$ de la variable aléatoire PT, lorsque celle-ci est prise comme exponentielle négative (Voir p.). Le format de lecture de AMU est F10.3.

e) AK, NAK

Le paramètre AK fournit la valeur du coefficient d'urgence K défini dans la relation de proportionalité

Le format de lecture correspondant est F5.1.

Le paramètre NAK fournit le nombre de répétitions d'une simulation donnée pour différentes valeurs de AK. Le format de lecture de NAK est I2.

NAK est utilisé comme paramètre d'une boucle DO d'exécution de la simulation.

Ainsi donc, après avoir fixé NAK, l'utilisateur doit fournir NAK valeurs de AK sur NAK cartes différentes.

f) IDISCPL

Le paramètre IDISCPL fixe la discipline d'ordonnancement désirée pour la simulation envisagée.

IDISCPL peut prendre les valeurs 1, 2, 3 avec la signification suivante:

Discipline	IDISCPL
Orientation	1
FCFS	2
SJFS	3

Table 2

Le format de lecture de IDISCPL est I1.

g) ICNTROL

Le paramètre ICNTROL fixe l'algorithme de commande choisi pour le traitement des files d'attente infaisables.

ICNTROL peut prendre les valeurs 1, 2 ou 3 dans le format I1 avec la signification suivante:

Algoritlme de commande	INCTROL
ALGDJ	1
ALGHJ	2

Table 3

h) IEND

Le paramètre IEND fixe la durée de la simulation qui s'arrête lorsque la valeur entière ICLOCK de l'horloge est supérieure à IEND.

L'organisation de la séquence de données pour RTSIM est donc la suivante

carte nº	Format	Paramètre
1	I1	IWRITE
2	I1	INTWRIT
3	F10.1,I3,F10.3	DT,M,ALAMDA
4	I1	IGENST
5	2F10.1 ou F10.3	TETAS,TETAM/AMU
6	I2	NAK
6+1	F5.1 } NAK	AK
6+NAK+1	I1	IDISCPL
6+NAK+2	I1	ICNTROL
6+NAK+3	I5	IEND

Table 4

FIGURE 1

'S DISCIPLINE

NEG EXPONENTIAL DISTRIBUTION

DT= [illegible] M= 16 K= 1.5 ALAMDA= [illegible]

TETA= 14.3

= 1.045

PARAMETERS TABLE

TETA	TAU	T	ISTATE
23.6	35.4	11.8	0
4.2	6.4	2.1	0
34.4	51.5	17.2	0
[illegible]	[illegible]	[illegible]	[illegible]
38.2	57.3	19.1	[illegible]
32.2	48.3	16.1	0
6.2	9.3	3.1	0
2.7	4.1	1.4	0
14.7	22.1	7.4	[illegible]
5.3	8.	2.7	0
1.9	2.9	1.0	0
1.5	2.2	[illegible].7	0
1.7	2.5	0.9	0
2.3	3.5	1.2	[illegible]
39.6	59.4	19.8	[illegible]
7.4	11.0	[illegible].7	0

IC

IENTER= 6 INTERRUPT ARRIVAL TIME TE(6)= [illegible].9

IENTER= 3 INTERRUPT ARRIVAL TIME TE(3)= 2.7
DROPPED JOB= 3 2.7

IENTER= 2 INTERRUPT ARRIVAL TIME TE(2)= 4.3

IENTER= 8 INTERRUPT ARRIVAL TIME TE(8)= 12.5

IENTER= 16 INTERRUPT ARRIVAL TIME TE(16)= 57.9

IENTER= 12 INTERRUPT ARRIVAL TIME TE(12)= 74.6

IENTER= 3 INTERRUPT ARRIVAL TIME TE(3)= 76.9

IENTER= 7 INTERRUPT ARRIVAL TIME TE(7)= 83.8

IENTER= 15 INTERRUPT ARRIVAL TIME TE(15)=122.3

IENTER= 4 INTERRUPT ARRIVAL TIME TE(4)=125.3

IENTER= 14 INTERRUPT ARRIVAL TIME TE(14)=152.3

IENTER= 5 INTERRUPT ARRIVAL TIME TE(5)=157.2

IENTER= 6 INTERRUPT ARRIVAL TIME TE(6)=162.3
DROPPED JOB= 6 162.3

IENTER= 6 INTERRUPT ARRIVAL TIME TE(6)=168.1
DROPPED JOB= 6 168.1

IENTER= 16 INTERRUPT ARRIVAL TIME TE(16)=216.1

IENTER= 13 INTERRUPT ARRIVAL TIME TE(13)=232.4

IENTER= 16 INTERRUPT ARRIVAL TIME TE(16)=274.7

IENTER= 8 INTERRUPT ARRIVAL TIME TE(8)=284.5

IENTER= 9 INTERRUPT ARRIVAL TIME TE(9)=294.1

IENTER= 13 INTERRUPT ARRIVAL TIME TE(13)=315.1

IENTER= 1 INTERRUPT ARRIVAL TIME TE(1)=315.4

IENTER= 7 INTERRUPT ARRIVAL TIME TE(7)=323.5

```
DROPPED JOB=          6.7
IENTER= 12  INTERRUPT ARRIVAL TIME   TE( 12)=356.1
IENTER= 14  INTERRUPT ARRIVAL TIME   TE( 14)=38[illegible]
IENTER=  9  INTERRUPT ARRIVAL TIME   TE(  9)=391.1
IENTER= 11  INTERRUPT ARRIVAL TIME   TE( 11)=407.2
IENTER=  6  INTERRUPT ARRIVAL TIME   TE(  6)=412.6
IENTER= 14  INTERRUPT ARRIVAL TIME   TE( 14)=436.7
IENTER= 15  INTERRUPT ARRIVAL TIME   TE( 15)=476.1
IENTER=  5  INTERRUPT ARRIVAL TIME   TE(  5)=481.4
DROPPED JOB=          481.4
IENTER=  4  INTERRUPT ARRIVAL TIME   TE(  4)=495.3
IENTER= 16  INTERRUPT ARRIVAL TIME   TE( 16)=515.3
IENTER=  7  INTERRUPT ARRIVAL TIME   TE(  7)=522.6
IENTER= 15  INTERRUPT ARRIVAL TIME   TE( 15)=556.2
IENTER=  9  INTERRUPT ARRIVAL TIME   TE(  9)=564.5
IENTER= 12  INTERRUPT ARRIVAL TIME   TE( 12)=583.7
IENTER= 14  INTERRUPT ARRIVAL TIME   TE( 14)=611.5
IENTER=  1  INTERRUPT ARRIVAL TIME   TE(  1)=612.3
IENTER=  8  INTERRUPT ARRIVAL TIME   TE(  8)=621.2
IENTER=  3  INTERRUPT ARRIVAL TIME   TE(  3)=624.0
IENTER=  5  INTERRUPT ARRIVAL TIME   TE(  5)=628.5
DROPPED JOB=          628.5
IENTER=  6  INTERRUPT ARRIVAL TIME   TE(  6)=634.9
DROPPED JOB=  6       634.9
IENTER=  2  INTERRUPT ARRIVAL TIME   TE(  2)=636.6
DROPPED JOB=  2       636.6
IENTER= 15  INTERRUPT ARRIVAL TIME   TE( 15)=672.9
IENTER=  5  INTERRUPT ARRIVAL TIME   TE(  5)=677.8
DROPPED JOB=  5       677.8
IENTER= 14  INTERRUPT ARRIVAL TIME   TE( 14)=702.0
IENTER= 13  INTERRUPT ARRIVAL TIME   TE( 13)=722.2
IENTER=  7  INTERRUPT ARRIVAL TIME   TE(  7)=728.9
IENTER=  8  INTERRUPT ARRIVAL TIME   TE(  8)=737.8
IENTER=  2  INTERRUPT ARRIVAL TIME   TE(  2)=739.0
IENTER=  3  INTERRUPT ARRIVAL TIME   TE(  3)=741.5
IENTER= 13  INTERRUPT ARRIVAL TIME   TE( 13)=761.7
IENTER= 14  INTERRUPT ARRIVAL TIME   TE( 14)=790.9
IENTER= 13  INTERRUPT ARRIVAL TIME   TE( 13)=813.6
IENTER= 14  INTERRUPT ARRIVAL TIME   TE( 14)=842.2
IENTER= 13  INTERRUPT ARRIVAL TIME   TE( 13)=864.0
IENTER= 15  INTERRUPT ARRIVAL TIME   TE( 15)=886.2
IENTER=  6  INTERRUPT ARRIVAL TIME   TE(  6)=902.2
DROPPED JOB= 15       902.2
IENTER=  3  INTERRUPT ARRIVAL TIME   TE(  9)=913.8
IENTER= 11  INTERRUPT ARRIVAL TIME   TE( 11)=930.9
DROPPED JOB= 11       930.9

IENTER= 12  INTERRUPT ARRIVAL TIME   TE( 12)=948.7
IENTER=  8  INTERRUPT ARRIVAL TIME   TE(  9)=957.9
```

FIGURE 2

IENTED DISCIPLINE

NEG EXPONENTIAL DISTRIBUTION

DT= 2.1 M= 15 K= 1.5 / LAMDA= 1.17

TETAM= 7.0

= 0.5

PARAMETERS TABLE

TETA	TAU	T	ISTATE
11.1	16.6	5.5	0
2.0	3.0	1.0	0
16.1	24.2	8.1	0
2.4	3.4	.9	0
17.9	26.8	8.9	0
15.1	22.7	7.6	0
2.9	4.4	1.5	0
1.3	1.9	0.6	0
6.9	10.4	3.5	0
2.5	3.7	1.2	0
0.9	1.4	0.5	0
0.7	1.0	0.3	0
0.8	1.2	0.4	0
1.1	1.6	0.5	0
18.6	27.8	9.3	0
3.4	5.2	1.7	0

DJ

IENTER= 6 INTERRUPT ARRIVAL TIME TE(6)= 2.1
QUEUE FEASIBLE

6	15.1	22.7	7.6	0	3

IENTER= 3 INTERRUPT ARRIVAL TIME TE(3)= 2.7
QUEUE INFEASIBLE

QUEUE STATE AND PARAMETERS WORK TABLE

JN	TETA	TAU	T	KFLAG	STATE(SUSP=1,INACT=2,ACT=3,TES
6	12.3	18.9	7.6	0	3
3	16.1	24.2	-4.2	1	6

JN	TETA	TAU	T	KFLAG	STATE(SUSP=1,INACT=2,ACT=3,TES
1	11.1	16.6	5.5	0	0
2	2.0	3.0	1.0	0	0
3	16.1	24.2	-4.2	1	6
4	2.4	3.4	.9	0	0
5	17.9	26.8	8.9	0	0
6	12.3	18.9	7.6	0	3
7	2.9	4.4	1.5	0	0
8	1.3	1.9	.6	0	0
9	6.9	10.4	3.5	0	0
10	2.5	3.7	1.2	0	0
11	0.9	1.4	0.5	0	0
12	0.7	1.0	0.3	0	0
13	.8	1.2	.4	0	0
14	1.1	1.6	0.5	0	0
15	18.6	27.8	9.3	0	0
16	3.4	5.2	1.7	0	0

DROP= 3
2.... 1

DROP= 6

1	0.000	6.579	8.051	0.000
2	6.579	7777.100	7777.[illegible]	3.632

1	1.000	0.100
2	0.000	0.100

0.000

DROPPED JOB= 6 2.7

3	16.1	24.2	10.9	0	3

```
 IENTER=  2  INTERRUPT ARRIVAL TIME   TE(  2)=  4.3
 QUEUE FEASIBLE

2        2.0          3.0          1.0     0          3
3       14.6         22.7          3.9     0          1
 HEAD JOB    2 COMPLETED      CLOCK=          6.3
3       14.6          .7          3.9                3

 IENTER=  8  INTERRUPT ARRIVAL TIME   TE(  8)= 12.5
 QUEUE FEASIBLE

3        1.3          1.3          .6                 3
3        8.3         14.4          7.6     0          1
 HEAD JOB    8 COMPLETED      CLOCK=         13.9
3        8.3         13.1          7.6     0          3
 HEAD JOB    3 COMPLETED      CLOCK=         22.3

 IENTER= 16  INTERRUPT ARRIVAL TIME   TE( 16)= 57.0
 QUEUE FEASIBLE

16       3.4          5.2          1.7                3
 HEAD JOB   16 COMPLETED      CLOCK=         57.6

 IENTER= 12  INTERRUPT ARRIVAL TIME   TE( 12)= 74.6
 QUEUE FEASIBLE

12        .7          1.          .3                 3
 HEAD JOB   12 COMPLETED      CLOCK=         75.3

 IENTER=  3  INTERRUPT ARRIVAL TIME   TE(  3)= 76.9
 QUEUE FEASIBLE

3       16.1         24.2          8.1     0          3

 IENTER=  7  INTERRUPT ARRIVAL TIME   TE(  7)= 83.8
 QUEUE FEASIBLE

7        2.9          4.4          1.5     0          3
3        9.3         17.4          5.1     0          1
 HEAD JOB    7 COMPLETED      CLOCK=         86.8
3        9.3         14.4          5.1                3
 HEAD JOB    3 COMPLETED      CLOCK=         96.2

 IENTER= 15  INTERRUPT ARRIVAL TIME   TE( 15)=122.3
 QUEUE FEASIBLE

15      18.6         27.8          9.3     0          3

 IENTER=  4  INTERRUPT ARRIVAL TIME   TE(  4)=125.3
 QUEUE FEASIBLE

 4        0.1          0.1          0.0     0          3
15       15.6         24.8          9.2     0          1
 HEAD JOB    4 COMPLETED      CLOCK=        125.4
15       15.6         24.7          9.2                3
 HEAD JOB   15 COMPLETED      CLOCK=        141.0

 IENTER= 14  INTERRUPT ARRIVAL TIME   TE( 14)=152.3
 QUEUE FEASIBLE

14        1.1          1.6          0.5     0          3
 HEAD JOB   14 COMPLETED      CLOCK=        153.5

 IENTER=  5  INTERRUPT ARRIVAL TIME   TE(  5)=157.2
 QUEUE FEASIBLE

 5       17.3         25.9          8.9     0          3

 IENTER=  6  INTERRUPT ARRIVAL TIME   TE(  6)=162.3
 QUEUE INFEASIBLE

 QUEUE STATE AND PARAMETERS WORK TABLE

  JN      TETA         TAJ           T       KFLAG     STATE(SUSP=1,INACT=2,ACT=3,TE
5        12.7         21.6          8.9     0          3
6        15.1         22.7         -5.1     1          6

  JN      TETA         TAJ           T       KFLAG     STATE(SUSP=1,INACT=2,ACT=3,TE
1        11.1         16.6          5.5     0          0
2         2.0          3.0          1.0     0          0
3        16.1         24.2          8.1     0          0
4         .1           .1           .0                 0
```

```
 5        12.7          31.9          -8.9      0          3
 6        15.1          33.7          -5.1      1          6
 7         2.9           4.4           1.5      0          0
 8         1.3           1.9           .6
 9         6.9          10.4           3.5                 0
10         2.6           3.7           1.2      0          0
11         0.3           1.4           0.5      0          0
12         0.7           1.1           0.3      0          0
13         0.9           1.2           .4                  1
14         1.1           1.6           .5                  1
15        18.6          27.8          9.3       0          0
16         3.4           5.2          1.7       0          0
  DROP=      6
       3.

  DROP=      5
    1       6.313         5.576         7.550        0.100
    2       5.576      7777.100      7777.000        1.629

    1       6.313         5.576

    2       0.000         0.100
         0.000

  DROPPED JOB=  5      162.3
 6        15.1          33.7          12.8      0          3

  IENTER=  7  INTERRUPT ARRIVAL TIME   TE(  7)=168.1
  QUEUE FEASIBLE

 7         2.9           4.4           1.5                 3
 6         9.4          17.0          9.8       0         1
  HEAD JOB    7 COMPLETED      CLOCK=      171.1
 5         9.4          14.0          9.8       0         3
  HEAD JOB    6 COMPLETED      CLOCK=      181.6

  IENTER= 16  INTERRUPT ARRIVAL TIME   TE( 16)=210.1
  QUEUE FEASIBLE

16         3.4           5.2          1.7                 3
  HEAD JOB   16 COMPLETED      CLOCK=      213.7

  IENTER= 13  INTERRUPT ARRIVAL TIME   TE( 13)=232.4
  QUEUE FEASIBLE

13         1.8           1.2           .4       0         3
  HEAD JOB   13 COMPLETED      CLOCK=      233.4

  IENTER= 16  INTERRUPT ARRIVAL TIME   TE( 16)=274.7
  QUEUE FEASIBLE

16         3.4           5.2          1.7       0         3
  HEAD JOB   16 COMPLETED      CLOCK=      278.2

  IENTER=  8  INTERRUPT ARRIVAL TIME   TE(  8)=234.5
  QUEUE FEASIBLE

 8         1.3           1.9           0.6      0         3
  HEAD JOB    8 COMPLETED      CLOCK=      285.9

  IENTER=  9  INTERRUPT ARRIVAL TIME   TE(  9)=294.1
  QUEUE FEASIBLE

 9         6.9          10.4          3.5                 3
  HEAD JOB    9 COMPLETED      CLOCK=      311.1

  IENTER= 13  INTERRUPT ARRIVAL TIME   TE( 13)=315.1
  QUEUE FEASIBLE

13          .8           1.2           .4       0         3

  IENTER=  1  INTERRUPT ARRIVAL TIME   TE(  1)=315.4
  QUEUE FEASIBLE

13          .5            .9           .4                 3
 1        11.1          15.6          5.0       0         2
  HEAD JOB   13 COMPLETED      CLOCK=      316.0
 1        11.1          15.6          5.0       0         3

  IENTER=  7  INTERRUPT ARRIVAL TIME   TE(  7)=323.5
  QUEUE FEASIBLE

 7         2.3           4.4          1.5       0         3
 1         3.5           8.4          2.1                 1
  HEAD JOB    7 COMPLETED      CLOCK=      326.6
```

```
  1
  2       0.000       0.100
      0.000

  DROPPED JOB= 15        481.4
 5        17.3        25.8        14.3                3

  IENTER=  4  INTERRUPT ARRIVAL TIME   TE(  4)=485.3
  QUEUE FEASIBLE

 4         1.1         1.1         0.0                3
 5        13.9        22.8        14.0     0          1
  HEAD JOB     4 COMPLETED      CLOCK=       485.5
 5        13.9        22.7        14.0     0          3
  HEAD JOB     5 COMPLETED      CLOCK=       490.4

  IENTER= 15  INTERRUPT ARRIVAL TIME   TE( 15)=515.3
  QUEUE FEASIBLE

15        10.6        27.1         9.3                3

  IENTER=  7  INTERRUPT ARRIVAL TIME   TE(  7)=522.6
  QUEUE FEASIBLE

 7         2.9         4.4         1.5                3
15        11.4        21.5         6.4                1
  HEAD JOB     7 COMPLETED      CLOCK=       525.6
15        11.4        17.5         6.4     0          3
  HEAD JOB    15 COMPLETED      CLOCK=       537.0

  IENTER= 16  INTERRUPT ARRIVAL TIME   TE( 16)=556.2
  QUEUE FEASIBLE

16         3.4         5.2         1.7                3
  HEAD JOB    16 COMPLETED      CLOCK=       559.8

  IENTER=  8  INTERRUPT ARRIVAL TIME   TE(  8)=564.5
  QUEUE FEASIBLE

 8         1.3         1.9         0.6                3
  HEAD JOB     8 COMPLETED      CLOCK=       565.8

  IENTER= 12  INTERRUPT ARRIVAL TIME   TE( 12)=583.7
  QUEUE FEASIBLE

12         0.7         1.0         0.3     0          3
  HEAD JOB    12 COMPLETED      CLOCK=       584.5

  IENTER= 14  INTERRUPT ARRIVAL TIME   TE( 14)=611.5
  QUEUE FEASIBLE

14         1.1         1.6         0.5     0          3

  IENTER=  1  INTERRUPT ARRIVAL TIME   TE(  1)=612.3
  QUEUE FEASIBLE

14         0.3         0.8         0.5     0          3
 1        11.1        16.6         5.2     0          2
  HEAD JOB    14 COMPLETED      CLOCK=       612.7
 1        11.1        16.7         5.2     0          3

  IENTER=  8  INTERRUPT ARRIVAL TIME   TE(  8)=621.2
  QUEUE FEASIBLE

 8         1.3         1.9         0.6     0          3
 1         2.6         7.8         4.0     0          1
  HEAD JOB     8 COMPLETED      CLOCK=       622.5
 1         2.6         6.5         4.1                3

  IENTER=  3  INTERRUPT ARRIVAL TIME   TE(  3)=624.0
  QUEUE FEASIBLE

 1         1.1         4.9         4.                 3
 3        16.1        24.2         7.1                2
  HEAD JOB     1 COMPLETED      CLOCK=       625.1
 3        16.1        23.2         7.1     0          3

  IENTER=  5  INTERRUPT ARRIVAL TIME   TE(  5)=629.5
  QUEUE INFEASIBLE

  QUEUE STATE AND PARAMETERS WORK TABLE

  JN      TETA      TAU         T       KFLAG    STATE(SUSP=1,INACT=2,ACT=3,TES
```

```
 9        6.9        17.4        3.5      0         3
 6        3.6        11.2        6.6                1
  HEAD JOB    9 COMPLETED      CLOCK=      921.8
 6        3.6        4.2         6.6      0         3
  HEAD JOB    6 COMPLETED      CLOCK=      924.5

  IENTER= 11  INTERRUPT ARRIVAL TIME   TE( 11)=93 .
  QUEUE FEASIBLE

11        0.9        1.4         0.5      0         3
  HEAD JOB   11 COMPLETED      CLOCK=      931.0

  IENTER= 12  INTERRUPT ARRIVAL TIME   TE( 12)=948.7
  QUEUE FEASIBLE

12        .7         1.          .5                 3
  HEAD JOB   12 COMPLETED      CLOCK=      949.4

  IENTER=  3  INTERRUPT ARRIVAL TIME   TE(  3)=957.9
  QUEUE FEASIBLE

 8        1.3        1.9         .6                 3
  HEAD JOB    8 COMPLETED      CLOCK=      959.2

  IENTER=  6  INTERRUPT ARRIVAL TIME   TE(  6)=964.0
  QUEUE FEASIBLE

 6       15.1        22.7        7.6      0         3
  HEAD JOB    6 COMPLETED      CLOCK=      979.3

  SIMULATION TIME=  1    .1    NUMBER ARRIVALS=    64   NUMBER DROPPED=   7
  MEAN INTERDROPPING TIME=    142.9    ACTUAL MEAN INTERARRIVAL TIME=     15.

IJ

  IENTER=  6  INTERRUPT ARRIVAL TIME   TE(  6)=  0.0
  QUEUE FEASIBLE
```

FIGURE 3

USER DISCIPLINE

MIG EXPONENTIAL DISTRIBUTION

D= 0.1 M= 16 K= 1.5 ALAMDA= 0.070

MEAN= 7.0

0.5

PARAMETER TABLE

ROW LOW F STATE

11.1	16.6	5.5	6
2.0	3.0	1.0	8
15.1	22.7	7.6	5
0.1	0.1	0.0	9
17.0	25.5	8.5	4
15.1	22.7	7.6	3
2.9	4.4	1.5	2
1.3	1.9	0.6	1
6.9	10.4	3.5	0
2.5	3.7	1.3	1
1.0	1.4	0.5	2
0.7	1.0	0.3	0
0.8	1.2	0.4	1
1.1	1.6	0.5	0
18.6	27.9	9.3	0
7.4	5.2	1.7	0

0

SIMULATION TIME= 1000.1 RUNS= 64 RUN NUMBER= 7
MEAN INTERARRIVAL TIME= 142.9 ACTUAL MEAN INTERARRIVAL TIME= 15.

0

SIMULATION TIME= 1000.1 RUNS= 64 RUN NUMBER= 7
MEAN INTERARRIVAL TIME= 142.9 ACTUAL MEAN INTERARRIVAL TIME= 15.

IS DISCIPLINE

0

SIMULATION TIME= 1000.1 RUNS= 0 RUN NUMBER= 1
MEAN INTERARRIVAL TIME= 66.7 ACTUAL MEAN INTERARRIVAL TIME= 15.

0

SIMULATION TIME= 1000.1 RUNS= 0 RUN NUMBER= 15
MEAN INTERARRIVAL TIME= 66.7 ACTUAL MEAN INTERARRIVAL TIME= 15

IS DISCIPLINE

0

SIMULATION TIME= 1000.1 RUNS= 64 RUN NUMBER= 7
MEAN INTERARRIVAL TIME= 142.9 ACTUAL MEAN INTERARRIVAL TIME= 15.

0

SIMULATION TIME= 1000.1 RUNS= 64 RUN NUMBER= 7
MEAN INTERARRIVAL TIME= 142.9 ACTUAL MEAN INTERARRIVAL TIME= 15.

DT= 0.1 K= 15 K= 7.0 ALPHA= 0.070

DELTA= 7.0

	11.1	77.4	66.3	
		13.	11.	
	1	11		
	0.1	0.6	0.5	
	17.	125.1	107.3	
	15.1	105.7	90.5	
	2.	20.4	17.5	
	1.	9.	8.	
	6.	48.	41.4	
	2.5	17.4	14.	
	0.	6.	5.4	
	0.7	4.	4.2	
	0.	2.	2.	
	1.1	7.7	6.6	
	16.6	129.	111.4	
	3.4	24.1	20.7	

SIMULATION TIME= 1000.1 NUMBER OF VALUES= 64 NUMBER DROPPED= 0
MEAN INTERCOMPLETION TIME= ACTUAL MEAN INTERCOMPLETION TIME= 15.

SIMULATION TIME= 1000.1 NUMBER OF VALUES= 64 NUMBER DROPPED= 0
MEAN INTERCOMPLETION TIME= ACTUAL MEAN INTERCOMPLETION TIME= 15.

FB DISCIPLINE

SIMULATION TIME= 1000.1 NUMBER OF VALUES= 64 NUMBER DROPPED= 0
MEAN INTERCOMPLETION TIME= 111.1 ACTUAL MEAN INTERCOMPLETION TIME= 15.

SIMULATION TIME= 1000.1 NUMBER OF VALUES= 64 NUMBER DROPPED= 0
MEAN INTERCOMPLETION TIME= 111.1 ACTUAL MEAN INTERCOMPLETION TIME= 15.

SJ DISCIPLINE

SIMULATION TIME= 1000.1 NUMBER OF VALUES= 64 NUMBER DROPPED= 0
MEAN INTERCOMPLETION TIME= ACTUAL MEAN INTERCOMPLETION TIME= 15.

SIMULATION TIME= 1000.1 NUMBER OF VALUES= 64 NUMBER DROPPED= 0
MEAN INTERCOMPLETION TIME= ACTUAL MEAN INTERCOMPLETION TIME= 15.

[illegible]

[illegible] EX[illegible] [illegible] DISTRIBUTION

D1= 0.1 [illegible] 16 K= 1.5 LAMBDA= 0.070

[illegible] 1= 14.1

1.0

[illegible]

TETA	[illegible]	T	TOTAL
22.7	33.[illegible]	11.1	0
4.[illegible]	6.6	2.[illegible]	0
32.4	4[illegible].[illegible]	16.2	0
[illegible].2	0.3	0.1	0
36.[illegible]	54.[illegible]	1[illegible].[illegible]	[illegible]
3[illegible].4	45.6	15.2	0
[illegible]	[illegible]	2.[illegible]	0
2.5	3.[illegible]	1.3	0
13.[illegible]	2[illegible].[illegible]	7.4	0
5.6	7.6	2.5	0
1.[illegible]	2.7	6.[illegible]	0
1.[illegible]	2.1	0.[illegible]	0
1.6	2.4	0.[illegible]	0
2.2	3.3	1.1	0
37.4	56.1	1[illegible].7	0
6.3	1[illegible].4	3.5	0

SIMULATION TIME= 1000.1   NUMBER ARRIVALS= 64   NUMBER DROPPED= [illegible]
MEAN INTERDROPPING TIME= 111.1   ACTUAL MEAN INTERARRIVAL TIME= 15.6

SIMULATION TIME= 1000.1   NUMBER ARRIVALS= 64   NUMBER DROPPED= 10
MEAN INTERDROPPING TIME= 100.0   ACTUAL MEAN INTERARRIVAL TIME= 15.6

[illegible] DISCIPLINE

SIMULATION TIME= 1000.1   NUMBER ARRIVALS= [illegible]   NUMBER DROPPED= 2
MEAN INTERDROPPING TIME= 41.7   ACTUAL MEAN INTERARRIVAL TIME= 15.6

SIMULATION TIME= 1000.1   NUMBER ARRIVALS= [illegible]   NUMBER DROPPED= 2
MEAN INTERDROPPING TIME= 41.7   ACTUAL MEAN INTERARRIVAL TIME= 15.6

[illegible] DISCIPLINE

SIMULATION TIME= 1000.1   NUMBER ARRIVALS= 64   NUMBER DROPPED= 10
MEAN INTERDROPPING TIME= 100.0   ACTUAL MEAN INTERARRIVAL TIME= 15.6

SIMULATION TIME= 1000.1   NUMBER ARRIVALS= 64   NUMBER DROPPED= 11
MEAN INTERDROPPING TIME= 90.9   ACTUAL MEAN INTERARRIVAL TIME= 15.6

1 2 [illegible]

[illegible]

[illegible] 0.1 [illegible] 15 K= 2.0 [illegible] 0.070

[illegible] 14.1

1 0

[illegible]

[illegible] [illegible] [illegible] [illegible]

22.3 44.5 22.3 0
0.5 1.1 5.0 1
32.4 64.7 32.4 0
1.2 2.3 6.2 0
36.1 72.1 36.0 0
35.6 66.7 36.4 0
5.3 11.7 5.3 0
2.6 5.4 2.5 0
13.9 27.7 13.8 0
5.0 10.0 5.0 0
1.1 2.3 1.2 0
1.4 2.9 1.4 0
1.5 3.0 1.5 0
2.2 4.4 2.2 0
3.1 6.2 3.1 0
6.9 13.7 6.9 0

J

[illegible] [illegible] 1000.1 [illegible] 64 [illegible]
[illegible] 125.0 [illegible] 15.6

J

[illegible] [illegible] 1000.1 [illegible] 64 [illegible]
[illegible] 71.4 [illegible] 15.6

2 0.000[illegible]

J

[illegible] [illegible] 1000.1 [illegible] 64 [illegible]
[illegible] 37.0 [illegible] 15.6

J

[illegible] [illegible] 1000.1 [illegible] 64 [illegible]
[illegible] 37.0 [illegible] 15.6

2 0.000[illegible]

J

[illegible] [illegible] 1000.1 [illegible] 64 [illegible]
[illegible] 125.0 [illegible] 15.6

J

[illegible] [illegible] 1000.1 [illegible] 64 [illegible]
[illegible] 71.4 [illegible] 15.6

1.0 SIMPLE

1.0 [illegible]

DT= 0.1 N= 16 K= 3.0 FLOW= 0.870

TIME= 14.1

1.0

[illegible]

22.3	[illegible]	[illegible]	[illegible]
4.1	12.3	7.3	[illegible]
32.4	7.3	64.7	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]
36.[illegible]	[illegible]	72.[illegible]	[illegible]
3[illegible].4	[illegible]	58.[illegible]	[illegible]
[illegible]	[illegible]	11.[illegible]	[illegible]
2.6	7.7	5.1	[illegible]
1[illegible]	41.7	27.[illegible]	[illegible]
5.[illegible]	15.[illegible]	1[illegible]	[illegible]
1.[illegible]	5.5	3.5	[illegible]
1.[illegible]	[illegible].2	2.[illegible]	[illegible]
1.6	4.[illegible]	3.[illegible]	[illegible]
2.2	6.6	4.4	[illegible]
37.4	112.[illegible]	74.[illegible]	[illegible]
6.[illegible]	[illegible].[illegible]	13.[illegible]	[illegible]

)

SIMULATION TIME= 1000.1 NUMBER OF VALUES= 64 NUMBER OF DISPOSE= 2

MEAN INTERVAL BETWEEN TIMES= 500.0 ACTUAL MEAN INTERVAL TIME= 15.6

)

SIMULATION TIME= 1000.1 NUMBER OF VALUES= 64 NUMBER OF DISPOSE= 4

MEAN INTERVAL BETWEEN TIMES= 250.0 ACTUAL MEAN INTERVAL TIME= 15.6

DISCIPLINE

)

SIMULATION TIME= 1000.1 NUMBER OF VALUES= 8 NUMBER OF DISPOSE= 23

MEAN INTERVAL BETWEEN TIMES= 43.5 ACTUAL MEAN INTERVAL TIME= 15.6

)

SIMULATION TIME= 1000.1 NUMBER OF VALUES= 8 NUMBER OF DISPOSE= 24

MEAN INTERVAL BETWEEN TIMES= 32.5 ACTUAL MEAN INTERVAL TIME= 15.6

DISCIPLINE

)

SIMULATION TIME= 1000.1 NUMBER OF VALUES= 64 NUMBER OF DISPOSE= 2

MEAN INTERVAL BETWEEN TIMES= 500.0 ACTUAL MEAN INTERVAL TIME= 15.6

)

SIMULATION TIME= 1000.1 NUMBER OF VALUES= 64 NUMBER OF DISPOSE= 2

MEAN INTERVAL BETWEEN TIMES= 500.0 ACTUAL MEAN INTERVAL TIME= 15.6

[illegible] EXAMPLE B

M/G EXPONENTIAL DISTRIBUT[illegible]

DT= 0.1 N= 16 K= 4.0 [illegible]= 0.070

TAU= 14.1

1.0

RAW MOMENTS TOTAL

DATA	MEAN	T	STATE
22.3	[illegible]	66.[illegible]	0
4.3	15.[illegible]	18.[illegible]	0
32.4	12.7	27.3	0
[illegible]	[illegible]	[illegible]	0
36.[illegible]	144.0	19[illegible]	0
30.6	121.7	[illegible]	0
5.[illegible]	23.6	17.5	0
2.6	10.[illegible]	7.7	0
13.[illegible]	55.6	1.[illegible]	0
5.[illegible]	21.0	15.[illegible]	0
1.[illegible]	7.3	[illegible]	0
1.4	5.[illegible]	4.3	0
1.6	6.5	4.[illegible]	0
2.2	[illegible]	6.6	0
3[illegible]	1.[illegible]	112.3	0
6.[illegible]	27.[illegible]	20.[illegible]	0

J

SIMULATION TIME= 1000.1 NUMBER OF ARRIVALS= 64 NUMBER OF DROPPED= 0
MEAN INTERARRIVAL TIME= [illegible] ACTUAL MEAN INTERARRIVAL TIME= 15.6

J

SIMULATION TIME= 1000.1 NUMBER OF ARRIVALS= 64 NUMBER OF DROPPED= 0
MEAN INTERARRIVAL TIME= [illegible] ACTUAL MEAN INTERARRIVAL TIME= 15.6

S DISCIPLINE

J

SIMULATION TIME= 1000.1 NUMBER OF ARRIVALS= 64 NUMBER OF DROPPED= 24
MEAN INTERARRIVAL TIME= 41.7 ACTUAL MEAN INTERARRIVAL TIME= 15.6

J

SIMULATION TIME= 1000.1 NUMBER OF ARRIVALS= 64 NUMBER OF DROPPED= 25
MEAN INTERARRIVAL TIME= 40.0 ACTUAL MEAN INTERARRIVAL TIME= 15.6

S DISCIPLINE

J

SIMULATION TIME= 1000.1 NUMBER OF ARRIVALS= [illegible] NUMBER OF DROPPED= 1
MEAN INTERARRIVAL TIME= 1000.1 ACTUAL MEAN INTERARRIVAL TIME= 15.6

J

SIMULATION TIME= 1000.1 NUMBER OF ARRIVALS= [illegible] NUMBER OF DROPPED= 1
MEAN INTERARRIVAL TIME= 1000.1 ACTUAL MEAN INTERARRIVAL TIME= 15.6

[illegible]

[illegible]

QT= [illegible] N= 10 **K= 1.5** [illegible] DA= 0.070

TOTAL= [illegible]

1.6

[illegible]

TIME	[illegible]	[illegible]	STATE
35.1	[illegible]	17.5	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]
4.1	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]
11.0	16.4	[illegible]	[illegible]

[illegible] 1000.1 NUMBER [illegible] 64 [illegible] 12
[illegible] 33.3 [illegible] 15.6

[illegible] 1000.1 NUMBER [illegible] 64 [illegible] 14
[illegible] 71.4 [illegible] 15.6

DISCIPLINE

[illegible] 1000.1 NUMBER [illegible] 64 [illegible] 31
[illegible] 32.3 [illegible] 15.6

[illegible] 1000.1 NUMBER [illegible] 64 [illegible] 32
[illegible] 31.3 [illegible] 15.6

DISCIPLINE

[illegible] 1000.1 NUMBER [illegible] 64 [illegible] 13
[illegible] 76.9 [illegible] 15.6

[illegible] 1000.1 NUMBER [illegible] 64 [illegible] 14
[illegible] 71.4 [illegible] 15.6

[illegible]

[illegible]

DT= 0.1 N= 16 K= 2.0 FL [illegible]= 0.070

[illegible]= 22.2

1.1

[illegible]

[illegible]

[illegible]	TAU	T	[illegible]
35.1	70.3	35.1	0
6.3	12.6	6.3	0
1.2	2.5	1.2	0
0.3	0.6	0.3	0
56.4	112.8	56.4	0
43.0	86.0	43.0	0
8.2	16.5	8.3	0
4.1	8.1	4.1	0
21.6	43.2	21.6	0
7.8	15.6	7.8	0
2.8	5.7	2.8	0
2.2	4.4	2.2	0
2.1	4.1	2.1	0
3.5	7.0	3.5	0
56.8	113.6	56.8	0
11.0	21.9	11.0	0

J

[illegible]= 1000.1 [illegible]= 64 [illegible]= 10

[illegible] 100.0 ACTUAL MEAN [illegible] 15.6

J

[illegible]= 1000.1 [illegible]= 64 [illegible]= 10

[illegible] 52.6 ACTUAL MEAN [illegible] 15.6

5 DESCRIBING

J

[illegible]= 1000.1 [illegible]= 64 [illegible]= 36

[illegible] 2.0 ACTUAL MEAN [illegible] 15.6

J

[illegible]= 1000.1 [illegible]= 0 [illegible]= 3

[illegible] 29.5 ACTUAL MEAN [illegible] 15.8

5 DESCRIBING

J

[illegible]= 1000.1 [illegible]= 64 [illegible]= 10

[illegible] 100.0 ACTUAL MEAN [illegible] 15.6

J

[illegible]= 1000.1 [illegible]= 64 [illegible]= 10

[illegible] 52.6 ACTUAL MEAN [illegible] 15.8

[illegible]

EXACT TOTAL DIMENSION

DT= 0.1 NS 16 K= 3.0 AL PHA= 0.070

T TA= 22.2

1 :

PARAMETRIC TOTAL

DATA TOT T STATI

35.1	1[illegible] 4	7[illegible].3	0
6.[illegible]	1[illegible]	12.6	[illegible]
51.2	15[illegible].5	102.3	0
0.3	[illegible]	6[illegible]	[illegible]
56.[illegible]	176.4	113.6	[illegible]
4[illegible].2	144.[illegible]	6[illegible]	[illegible]
[illegible].4	27.[illegible]	11.5	[illegible]
4.1	13.2	[illegible].1	0
21.[illegible]	[illegible].[illegible]	[illegible].3	[illegible]
7.[illegible]	23.7	15.[illegible]	[illegible]
[illegible].[illegible]	[illegible].6	5.7	[illegible]
[illegible].[illegible]	6.5	[illegible].4	[illegible]
[illegible].[illegible]	7.7	5.1	[illegible]
[illegible].[illegible]	10.[illegible]	[illegible].0	[illegible]
50.[illegible]	177.[illegible]	11.[illegible]	[illegible]
11.0	32.[illegible]	21.[illegible]	0

J

STANDARD DEVIATION= 1000.1 NUMBER OF VALUES= 64 NUMBER OF DROPPED= 6
MEAN INTERVAL BETWEEN PEAKS= 166.7 ACTUAL MEAN INTERVAL TIME= 15.6

J

STANDARD DEVIATION= 1000.1 NUMBER OF VALUES= 64 NUMBER OF DROPPED= 12
MEAN [illegible] BETWEEN PEAKS= 25[illegible].3 ACTUAL MEAN INTERVAL TIME= 15.6

[illegible] DISCRIPLINE

J

STANDARD DEVIATION= 1000.1 NUMBER OF VALUES= 64 NUMBER OF DROPPED= 31
MEAN INTERVAL BETWEEN PEAKS= 32.3 ACTUAL MEAN INTERVAL TIME= 15.6

I

STANDARD DEVIATION= 1000.1 NUMBER OF VALUES= 64 NUMBER OF DROPPED= 3[illegible]
MEAN [illegible] BETWEEN PEAKS= 25.6 ACTUAL MEAN [illegible] TIME= 15.6

[illegible] DISCRIPLINE

J

STANDARD DEVIATION= 1000.1 NUMBER OF VALUES= 64 NUMBER OF DROPPED= 7
MEAN INTERVAL BETWEEN PEAKS= 142.[illegible] ACTUAL MEAN INTERVAL TIME= 15.6

I

STANDARD DEVIATION= 1000.1 NUMBER OF VALUES= 64 NUMBER OF DROPPED= 11
MEAN [illegible] BETWEEN PEAKS= 20.[illegible] ACTUAL MEAN INTERVAL TIME= 15.6

```
DISCOVERY
   AVG. EXPLORATION & DEVELOPMENT
   DT=      0.1   TF= 16   K=  4.0   ALPHA=    0.070
   TOTAL= 32.2
  1.5

   EXPLORATION DATA

      TIME        TAU              T        STATE

    35.1        140.6          135.4          0
     6.2         25.2           19.1          0
     1.5         23.1            1            0
     0.3          1.1            0.2          0
    56.4        227.3          170.4          0
    41.0        151.9          144.0          0
     5.3         32.1           27.1          0
     4.1         16.2           12.2          0
    21.3         87.3           66.1          0
     7.4         31.6           23.2          0
     2.8         11.5            8.8          0
     2.2          9.2            6.5          0
     2.7         10.2            7.5          0
     3.5         13.9           10.5          0
    59.0        236.0          177.0          0
    11.0         43.9           32.9          0
J

   SIMULATION TIME=   1000.1    NUMBER OF ACTIVITIES=   64   NUMBER OF QUEUES=   3
   MEAN QUEUE LENGTH=    530.9   ACTUAL MEAN INTERARRIVAL TIME=    15.6
J

   SIMULATION TIME=   1000.1    NUMBER OF ACTIVITIES=   64   NUMBER OF QUEUES=   6
   MEAN QUEUE LENGTH=    125.9   ACTUAL MEAN INTERARRIVAL TIME=    15.6
$ DISCIPLINE
J

   SIMULATION TIME=   1000.1    NUMBER OF ACTIVITIES=   64   NUMBER OF QUEUES=  27
   MEAN QUEUE LENGTH=     31.0   ACTUAL MEAN INTERARRIVAL TIME=    15.6
J

   SIMULATION TIME=   1000.1    NUMBER OF ACTIVITIES=   64   NUMBER OF QUEUES=   3
   MEAN QUEUE LENGTH=     27.4   ACTUAL MEAN INTERARRIVAL TIME=    15.6
$ DISCIPLINE
J

   SIMULATION TIME=   1000.1    NUMBER OF ACTIVITIES=   64   NUMBER OF QUEUES=   3
   MEAN QUEUE LENGTH=    332.4   ACTUAL MEAN INTERARRIVAL TIME=    15.6
J

   SIMULATION TIME=   1000.1    NUMBER OF ACTIVITIES=   64   NUMBER OF QUEUES=   7
   MEAN QUEUE LENGTH=    142.2   ACTUAL MEAN INTERARRIVAL TIME=    15.6
```

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

FA[illegible]

TOTA[illegible]

35.1	175.7	140.6	0
6.3	31.6	25.2	0
21.2	[illegible]	[illegible]	[illegible]
0.3	1.4	1.1	0
56.[illegible]	2[illegible].1	22[illegible].[illegible]	[illegible]
4[illegible].[illegible]	2[illegible].[illegible]	1[illegible].[illegible]	0
[illegible].[illegible]	46.[illegible]	37.[illegible]	[illegible]
4.1	20.[illegible]	16.[illegible]	[illegible]
21.[illegible]	10[illegible].7	[illegible].7	[illegible]
7.[illegible]	[illegible].5	71.[illegible]	0
2.[illegible]	16.4	11.[illegible]	[illegible]
2.2	11.0	[illegible].[illegible]	[illegible]
2.[illegible]	12.[illegible]	10.3	0
3.[illegible]	17.4	13.[illegible]	[illegible]
59.[illegible]	2[illegible].4	236.9	0
11.[illegible]	54.[illegible]	43.[illegible]	0

J

[illegible]

J

[illegible]

2 [illegible]

J

[illegible]

J

[illegible]

3 [illegible]

J

[illegible]

J

[illegible]

[illegible]

[illegible]

DT= 0.1 TM= 15 PA= 6.3 'L'D'= 0.070

T_TAP= 22.2

1 :

[illegible]

TSTA 1/4 T 'STATE

35.1 21.0 125.7 0
6.3 37.0 31.5 0
11.2 20.0 0 0
0.3 1.7 1.4 0
56.5 34.0 2 6.1 0
4 [illegible] 2.7 [illegible] 0
[illegible] 55.6 46.3 0
4.1 24.3 23.2 0
21. [illegible] 161.6 [illegible] 0
7. [illegible] 47.4 [illegible] 0
2. [illegible] 17.2 [illegible] 0
2.2 13.2 11.0 0
2. [illegible] [illegible] 12. [illegible] 0
3.5 20. [illegible] 17.4 0
5. [illegible] 33.3 2 4. [illegible] 0
11.0 65.7 54.7 0

DJ

STIMULATION TIME= 1000.1 NUMBER OF VALUES= 64 NUMBER OF DROPPED= 0
MEAN INTER-STIMULUS TIME= [illegible] ACTUAL MEAN INTERVAL TIME= 15.6

DJ

STIMULATION TIME= 1000.1 NUMBER OF VALUES= 64 NUMBER OF DROPPED= 0
MEAN INTER-STIMULUS TIME= [illegible] ACTUAL MEAN INTERVAL TIME= 15.6

'S DESCRIPTION

DJ

STIMULATION TIME= 1000.1 NUMBER OF VALUES= 64 NUMBER OF DROPPED= 21
MEAN INTER-STIMULUS TIME= 4.6 ACTUAL MEAN INTERVAL TIME= 15.6

DJ

STIMULATION TIME= 1000.1 NUMBER OF VALUES= 64 NUMBER OF DROPPED= 32
MEAN INTER-STIMULUS TIME= 31.3 ACTUAL MEAN INTERVAL TIME= 15.6

'S DESCRIPTION

DJ

STIMULATION TIME= 1000.1 NUMBER OF VALUES= 64 NUMBER OF DROPPED= 2
MEAN INTER-STIMULUS TIME= 59.0 ACTUAL MEAN INTERVAL TIME= 15.6

DJ

STIMULATION TIME= 1000.1 NUMBER OF VALUES= 64 NUMBER OF DROPPED= 2
MEAN INTER-STIMULUS TIME= 125.0 ACTUAL MEAN INTERVAL TIME= 15.6

REFERENCES

(1) "Theory of scheduling". R.W. Conway, W.L. Maxwell, L.W. Miller Addison-Wesley Publishing Company 1967.

(2) "Description et Mode d'emploi d'Evariste (Moniteur de multiprogrammation en temps réel)". J.P. Gouyon. Publication Nº 627. Dec. 1969.
L.A.A.S. du CNRS, Toulouse.

(3) "Optimal Scheduling of Linked repetitive Data-driven Processes". H.H. Johnson, Department of Computing and Control, Imperial College of Science and Technology, London SW7 2BT.

(4) "Single server queues with known service times and specified times for completion of service". H.H. Johnson, Department of Computing and Control, Imperial College of Science and Technology, London SW7 2BT.

(5) "A Real-time Operating System for Manned Spaceflight". P.W. Weiler, R.S. Kopp, R.G. Dorman, IEEE Transactions on Computers, Vol. C.19, Nº 5, May 1970.

(6) "Interacting Parallel Processes in a Real Time Environment". H.H. Johnson, Department of Computing and Control, Imperial College of Science and Technology, London SW7 2BT.

(7) "Some principles of time-sharing scheduler strategies" H. Hellerman. IBM Syst.J Nº 2, 1969.

(8) "Virtual systems queueing"
P.A. Johnson. Computing 29.11.73 The British Computer Society.

(9) "An Introduction to Probability Theory and its Applications"
W. Feller Vol. 1

(10) "A Queueing Model Of a Multiprogrammed Computer with a two level Storage System"
G.S. Shedler. Communications of the ACM, Jan 1973 vol. 16 Nº1

(11) "Mathematical Theory of Reliability".
R.E. Barlow and F. Proschan. The SIAM Series in applied Mathematics.

(12) "Queues, Inventories ans Maintenance".
P.M. Morse. MIT. Publications in Operation Research.

(13) "Scheduling with Deadlines ans loss Functions"
R. Mc Naughton. Management Science, Oct 1959 vol 6 Nº1.

(14) "Scheduling Alforithms for Multiprogramming in a Hard Real Time Environment".
C.L. LIU and J.W. LAYLAND. JACM Jan 1973 Vol. 20 Nº 1.

(15) "Analysis and Optimization of a Queuing model of a Real Time Computer Control System"
L. Schrage. IEEE Transaction on Computer. Nov 1969 Vol C18 Nº 11

(16) "Algèbre Moderne et Théorie des graphes".
B. Roy.Finance et Economie Appliquée.

(17) "Ordonnancement des processus temps réel sur une ressource préemptive".
J. Labetoulle. IRIA Rapport de Recherche nº 74, Mai 1974.

(18) "Discrete State Markov Processes"
M. Howl. Department of Computing and Control, Imperial College of Science and Technology. London SW7 2BT

(19) "Queueing Theory"
D.R. Cox

(20) "A Shortest Deadline Scheduling Algorithm for a Single Computer System in a Hard-Real Time Environment"
D. Julvé and H.H. Johnson
Imperial College of Science and Technology London SW7