JUL 7 1966

MASTER

COO-614-45

ICR QUARTERLY REPORT

Number 9
May 1, 1966

REPLACES FOR ANNOUNCEMENT
IN NUCLEAR SCIENCE ABSTRACTS

THE UNIVERSITY OF CHICAGO

THE INSTITUTE FOR COMPUTER RESEARCH

DISCLAIMER

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

DISCLAIMER

Portions of this document may be illegible in electronic image products. Images are produced from the best available original document.

N.C. \$ 5.00; MN 1.00 COO-614-45

The Institute for Computer Research

The University of Chicago

RELEASED FOR ANNOUNCEMENT
IN NUCLEAR SCIENCE ABSTRACTS

ICR Quarterly Report

Number 9
May 1, 1966

LEGAL NOTICE

This report was prepared as an account of Government sponsored work. Neither the United States, nor the Commission, nor any person acting on behalf of the Commission:

A. Makes any warranty or representation, expressed or implied, with respect to the accuracy, completeness, or usefulness of the information contained in this report, or that the use of any information, apparatus, method, or process disclosed in this report may not infringe privately owned rights; or

B. Assumes any liabilities with respect to the use of, or for damages resulting from the use of any information, apparatus, method, or process disclosed in this report.

As used in the above, "person acting on behalf of the Commission" includes any employee or contractor of the Commission, or employee of such contractor, to the extent that such employee or contractor of the Commission, or employee of such contractor prepares, disseminates, or provides access to, any information pursuant to his employment or contract with the Commission, or his employment with such contractor.

Prepared under the auspices of AEC Contract No. AT(11-1)-614

Richard H. Miller, Director of the Institute for Computer Research
and Principal Investigator

Table of Contents

Section I

- A. Maniac III Engineering Activity
- B. Data Transmission Between the Maniac III and the IBM 7040 (R. M. Herwitt, J. Bounin and C. K. Teng)
- C. The Man-Machine Report (C. C. J. Roothaan)

Section II

- A. M -Capture Program (K. Laitinen, N. Alton and J. Klems)
- B. High-Energy Inter-Nuclear Cascades (A. Van Ginneken et al.)

Section III

- A. On the Convergence of an Accelerated Iterative Method in the Solution of Singular Equations (W. V. Petryshyn)
- B. Further Remarks on Nonlinear P-Compact Operators in Banach Space (W. V. Petryshyn)
- C. Polarization in Stellar Dynamical Systems (R. H. Miller)
- D. Relaxation in a Model Cluster (R. H. Miller)
- E. Some Comments on Gilbert's Analytic Grammar (C. L. Robinson)
- F. The Production and Stabilization of Real-Time Task Schedules (G. K. Manacher)

Staff of the Institute for Computer Research

Naomi Alton¹

Robert Ashenhurst

Jurgen Bounin

Dolores Brennan

Kenneth Burns

Rachel Chakrin

Elaine Chow

Stephen Coyne

David Eicher

Abel Felix

Charles Fischer

William Georgas

Robert Graves

Richard Herwitt

Leonardo Herzenberg

Valerie Hillier

Paul Kosinski

Hirondo Kuki

George Kreisheimer

Vincent Kruskal

Glenn Manacher

Richard Miller

William Mixon

Lily Monheit

John Munn

Michael Neumann

Daniel Norman

Alex Orden

Walter Petryshyn

Charles Robinson

Clemens Roothaan

Mary Ruel

Lyle Settle

John Shepherd

Ilene Sievert

Albert Steiner

Cheng-kiat Teng

Serena Torres

Nancy Trombetta

Linda Uhler

Michael Williams

MANIAC III ENGINEERING ACTIVITY

1. Connection to IBM 7040 Computer.

This work has been completed except for final testing which requires allocation of 7040 machine time. Only a few minutes of such time has been made available and work is practically at a standstill.

It has been decided to use this link to connect Maniac to the 2701 Data Adaptor of the IBM 360/30 now being installed in Maniac Computer Room. Only trivial modifications are necessary, all at the Maniac end of the link.

Besides simplifying and accelerating the work of debugging the Maniac-IBM link this decision gives the additional benefit that the first of the interface units need not be assigned to the control of the IBM 360/30. This will avoid delay in releasing the Univac 1004 and allow more time for perfecting the design of the interface unit.

2. Interfacing Units for Future Peripherals.

a) A number of minor problems has delayed fabrication of the printed circuit boards, especially the large digit boards and logic boards, for this project. The first samples of the digit boards have been received and are now being tested.

b) The Maniac program simulating the performance of an interface unit caused some programmers to become interested in the logical design details of the interface units. As a result of

discussion some improvements have been agreed upon which, owing to delay in delivery of the printed circuit boards, it is not too late to incorporate in the design.

c) A Maniac program has been written to facilitate the preparation of chassis-wiring lists. With its aid, wiring lists for all the repetitive wiring (about 90%) in the digit board chassis have been prepared and this part of the wiring has been completed on one chassis.

d) Design of the logic printed-circuit boards has been completed-- delivery of sample boards is awaited. These will be large boards of the same size as the digit boards. Work on the design of the logic for internal routines has been suspended during discussion with programmers; however previous work indicates that these can be compactly laid out on logic boards. As an example one logic board would contain all the following routines:

Instruction Fetch

2. Address Calculation Routines

Operand Fetch

Operand Store

e) The 4-phase clock referred to in Quarterly Report No. 8 is a constant-frequency device. A variable-frequency version, which will be more useful for testing and evaluation of circuits, is under construction.

3. Queueing Control for Interface Units.

The general design is complete and is now being translated into terms of circuit packages. The queueing control will be built up entirely of the standard circuit cards used in the interface units.

4. Integrated Circuit Package Tester.

A "GO: NO-GO" testing unit for integrated circuit packages is being built. It is based on comparing output waveforms from the device under test with a standard device of the same type which is known to be good. It is adapted from a design of Burgmeier and Thom.¹

5. Reserved Addresses

The Maniac reserved address codes were changed as planned without difficulty.

There have been no major engineering shut-downs during this period.

¹"An Integrated Circuit Tester" J. C. Burgmeier and D. G. Thorn, EDN, March, 1966.

MANIAC III USE

January 28 - April 28, 1966

(Programmer and production use, debugging and production times are not separated in this table).

Engineering Computer Time	96.0
Orthogonal Latin Squares (Manacher)	91.3
Systems Work	65.7
Student Programming Experiments	44.8
Astronomical Calculations (Miller)	41.9
Numerical Experiments (Ashenhurst)	11.1
Miscellaneous	55.0

The student Programming Experiments result from an offer whereby Maniac III is made available to undergraduates who would like to try a computer program. It is an experimental program that at once permits systems programs to be tested in a non-critical environment and may help to develop good programming talent.

The Engineering Computer time was principally used in developing and operating simulators for the multi-computer system, in generating wiring tables for the interface controllers, and so on. Very little was used on engineering tests of Maniac III itself.

DATA TRANSMISSION BETWEEN THE MANIAC III AND THE IBM 7040

R. M. Herwitt, J. Bounin and C. K. Teng

The 7040 Subcomputer provides a path for data and control information travelling from Maniac III to the IBM 7040 and back. A direct data transmission path to the 7040 is an indirect path to the IBM 7094, since the 7040 coordinates the input and output activities of the 7094. Consequently, the Subcomputer enables Maniac III programmers to use the computing and input-output facilities of the Computation Center without carrying decks of cards from one installation to the other. What jobs will be performed in the Computation Center and what codes will request these jobs will be established by systems programming.

The transmission of data to or from the 7040 is an operation with five stages. The first stage is performed by Maniac III, which delegates full responsibility for the rest of the operation to the Subcomputer and 7040 at the end of the stage. The Maniac III instruction specifies two sets of memory locations. The first is the set of locations from which data will be taken or into which it will be placed, and its leftmost location is specified by the right designator of the instruction; the second set, consisting of only two locations, has the control word, specified by the left designator, and a location reserved for the report word immediately to the right of that. The control word contains job specification codes, to be established by the systems programmers, and the report word consists of information describing

the conditions that occurred during the operation and is made up partly by the 7040 program and partly by the Subcomputer circuitry. These formats are listed in the Appendix.

In executing the first stage of the operation, Maniac III does three things:

- a) it places the two designators into the Buffer Register so that they are accessible to the Subcomputer;
- b) it writes a full word of zeros into the location reserved for the report word, so that the programmer can tell that the operation has finished from the fact that this location no longer has all zeros in it;
- c) it starts the Subcomputer;
- d) it continues its program.

In the second stage of the operation, the Subcomputer does four things:

- a) it takes the two designators left in the Buffer Register by Maniac III, and places them into its own two address registers which are named DAR (Data Address Register) and CWAR (Control Word Address Register);
- b) it sends the Attention signal to the 7040, which causes the 7040 to interrupt its program and to execute the program that handles Maniac III affairs;
- c) it waits for the Attention Response signal from the 7040, which signifies that the 7040 has taken notice of the Attention signal and is ready to transact business with the subcomputer;
- d) it suspends its activities and waits for the next command from the 7040.

The writing of data is initiated by a Write command, which causes the Subcomputer to do two things:

- a) it takes a data word offered to it by the 7040, places it into the memory, and repeats these two actions on all words subsequently offered to it, stopping when it receives a Stop command;
- b) it suspends further activity until the next command arrives from the 7040.

Since the 7040 has a 36-bit word and the Ampex memory has a 48-bit word, and since the Subcomputer stores only full 48-bit words into the memory, a partial word is left if the 7040 does not send an integral multiple of 48 bits; this partial word will not be stored in the memory.

The last stage of the operation begins when the Subcomputer receives a Control command from the 7040. During this stage the Subcomputer does four things:

- a) it takes the report word, 36 bits long, offered to it by the 7040;
- b) it adds 12 bits of its own to the report word, thus describing the conditions it has detected;
- c) it stores the 48-bit word thus constructed into the memory;
- d) it stops completely.

This general description of the operation of the 7040 Subcomputer must necessarily be modified to accommodate errors and unusual conditions. The main idea is as given, however, and the details that qualify it will appear in the forthcoming manual, "The Operation of the 7040 Subcomputer."

Appendix

Formats

The Maniac III instruction format is the following:

Op. Code	δ^L	δ^R
7a	Control word address	Leftmost data address

The inflector bits are not used and may contain any value.

The memory words whose addresses are δ^L and $\delta^L + 1$ have the following formats;

δ^L	$\delta^L + 1$	
Control word	2^{47} through 2^{12}	- program re- port word
	2^{11} - ATT	these identify the stage at which the operation stopped.
	2^{10} - READ	
	2^9 - CW	
	2^8 - WRITE	
	2^7 - REP	
	2^6 - DT	this states whether any data was trans- mitted
	2^5 - END	
	2^4 - UNEND	these identify the reason for stop- ping
	2^3 - TIMEOUT	
	2^2 - STOP	
	2^1 and 2^0	not used (always equal to $(01)_2$)

All addresses should refer to Ampex memory locations, since the

Subcomputer cannot gain access to the Maniac III memory.

THE MAN-MACHINE REPORT*

C. C. J. Roothaan

Progress Report No. 5

(for the period 1 January 1966 through 31 March 1966)

1. 7040 Control Program

- a. Coding of the system scheduler by Mr. Kruskal is still in progress. Although the basic specifications for the scheduler have not been changed, some modifications and extensions of the original design have been made, particularly in regard to job setup scheduling. The principal extension involves a separate setup scheduler which will operate, to some extent, independently of the actual job input scheduler.
- b. General agreement was finally reached between the Computation Center and the Biological Sciences Computation Center on programming specifications for the interface between the 7040 Control Program and the B. S. C. C. 1401. According to the specifications, job input and output between the 7040 and 1401 will be scheduled and processed in essentially the same fashion as job input and output between the 7040 and its own readers, printers, and punches. In addition, the 1401

*Work on this project, supported by a grant from the International Business Machines Corporation with C. C. J. Roothaan as principal investigator, is carried on under the joint auspices of the Institute and the Computation Center.

operators will be afforded essentially the same control over system task scheduling as will be available to 7040 operators.

- c. Mr. Kosinski has incorporated two features into the multi-programming supervisor. The first is designed to resolve any conflicts arising from simultaneous references by independent programs to a common data block. The second feature is concerned with the allocation of resources, particularly memory segments to serve as I/O buffers, etc. An overlapped I/O process now does not use straightforward single or double buffering, but rather pooled buffering. With this scheme, a memory segment is requested immediately before a buffer is needed, and the segment is returned to the pool immediately after use. Since buffers are no longer assigned permanently, but circulate freely under control of the memory segment allocator, graceful degradation of system performance becomes possible. That is, when more buffers are needed than are available, all I/O processes are slowed partially, rather than some being stopped entirely due to lack of buffers. Thus the allocator, like the system scheduler, aims at acceptable performance for as many users as possible, rather than optimal performance for a few.
- d. Miss Monheit is currently investigating the feasibility of modifying the 7040 system assembler to automatically produce

fixed-length control sections. This modification, coupled with a suitable system loader and the newly-coded memory segment allocator, would allow the proposed utility routines, as well as the assembler itself, to be loaded into whatever segments happen to be available at load-time. This segmented assembly and loading scheme, somewhat reminiscent of various "paging" techniques, would afford more flexible and efficient resource allocation for the Control Program.

- e. Mr. Williams has completed coding of the job deck processing module for the Control Program. The module is a single re-entrant program which is capable of simultaneous processing of job input from all sources for which appropriate select routines are provided. Initially, the module will accept job input from the two 7040 card readers and the 1401 card reader at the Biological Sciences Computation Center. In preliminary tests, the module was found to drive both 7040 card readers simultaneously at full speed.
- f. All necessary hardware has arrived for the connection between the 7040 and the Maniac III computer at the Institute for Computer Research. Hardware debugging for the link has begun. Specifications for the nature and extent of the tie-in are currently being set. It is expected that the 7040-Maniac link will be used principally to render 7040 peripheral equipment directly available to the Maniac III.

2. 7040/7094 Interface

- a. The difficulties encountered in adapting the program-switchable tape drive feature as an input/output transmission medium remain unresolved. Several schemes for circumventing the problems have been proposed by system programmers and customer engineers, but all solutions have so far been found unworkable on grounds of either hardware difficulties or programming requirements. While some hope remains that a solution may yet be found, thought is being given to the possibility of re-writing existing input/output routines to use direct data link between the machines as a direct medium for transmission of job input and output. This proposal had been considered earlier in the development of the Dual Machine System, and fairly extensive preliminary work on the project had begun. The proposal was subsequently laid aside in favor of the tape-switching scheme because preliminary hardware specifications for the switchable drives indicated that tape-switching could achieve nearly the same performance level as "pure" direct data with substantially less programming expense.
- b. Mr. Steiner has begun writing a new version of SIGN-ON, the 7094 FORTRAN Monitor System record summoned for job-to-job transition. The new version will incorporate conventions established for the new 7040 Control Program. Due to the difficulties mentioned above, the current effort must be limited to those sections of SIGN-ON which do not depend directly on the nature of the input/output

transmission medium.

3. 7094 FORTRAN Monitor System

A version of COMIT, the compiler and interpreter developed at the Massachusetts Institute of Technology, has been made available to users under the U. of C. version of the FORTRAN Monitor System. COMIT is a programming language designed for non-numerical computer applications such as algebra, theorem-proving, and language translation.

μ -CAPTURE PROGRAM

K. Laitinen, N. Alton, J. Klems

A program to process data gathered from studies of the basic capture process

has been written for the CDC 3600 computer at Argonne National Laboratory. Using two or three cameras mounted on the front window of the chamber, photographs were taken of the reaction (1) as an $\sim 99\frac{1}{2}\%$ pure muon beam from the cyclotron was brought to rest in an 8-liter hydrogen bubble chamber. The reaction is identified by the appearance of a recoil proton created by the neutron and by the absence of a decay electron. The resulting photographs are scanned manually and points selected from each view are digitized and punched onto paper tape, the information from related views making up one frame of data. The paper tape data is then transferred to magnetic tape and used as input data to the program.

For this program a track may be a singular point alone, called an E event, a short two-point recoil proton track, called a P event, or a curved track of three or more points, called a C event. The program constructs three-dimensional coordinates for each track point, associates views and tries to match events by finding corresponding points in the several views. It computes the energies of the neutron

II A-2

which produced the recoil and of the proton, the path lengths of the neutron and proton, the radius, depth and the quantity Q , where Q represents the difference between measurements in two views.

The following parameters are used as input to the program:

α_i and β_i - cosine and sine, respectively, of the angle between the lines joining any i^{th} camera axis with the other two axes.

a_i and b_i - components in cm. of the vector $\vec{AB}$ in the right hand coordinate of the appropriate axis.

h - correction for glass and contents of bubble chamber index of refraction

X_o, Y_o, Z_o - center of sensitive volume

K - density of hydrogen

$(e, D, R)_{\text{max}}$ - maximum allowable value for proton and neutron energy, depth and radius respectively, for magnetic tape output

$(\text{option})e, p, c$ - parameters which give the option of omitting, or processing the E, P or C events respectively.

Under normal conditions they should always assume a value ≥ 2 .

A_x^i, A_y^i - real space coordinates in cm. of the fiducial mark A in the right hand coordinate system associated with each camera axis

d_i - distance between the optical axis of cameras i and $i-1$.

δ - parameter used to define closeness of fit

s - scale factor used to convert input coordinates (from measuring table tape) to centimeters.

Coordinate Systems

In the processing of data six coordinate systems are used, three of which are left-handed and three right-handed; one left-handed and one right-handed system is centered at each of the points I, II, III, (see Figure 1) which are the intersections of the camera axes with the front glass of the bubble chamber. For both right-handed and left-handed systems the z-axis is taken to point into the chamber.

Let us call the points I, II, III the first, second and third origin, since each point is to be the origin for two coordinate systems. The left-handed, or principal, coordinate system for the i^{th} origin is then defined by taking as the x-axis the line joining the i^{th} origin with the origin following it in cyclic permutation; for example, for the first origin the x-axis is taken along the line joining the first and second origin. The y-axis is then defined from the x- and z-axes and the requirement of left-handedness.

For ease in comparison of points in different views, it is convenient to define a second coordinate system for each origin. This system, denoted the rotated system, is right-handed and is defined for the i^{th} origin by taking as the x'-axis the line joining the i^{th} origin with the origin following it in anticyclic permutation. Thus, for the first origin the x'-axis would be the line joining points I and III. The y-axis is defined by right-handedness. It can easily be seen

II A-4

that with these definitions the comparison of points between two views is greatly simplified by using the principal coordinates of the points in one view and the rotated coordinates of the points in the other. The two sets of coordinates can then be transformed into the same system by changing the sign of the x-coordinates of one set and performing a simple displacement in the x-direction.

The points marked A and B in the figure are the front glass fiducials. These make possible the correlation of the coordinate systems with the actual bubble-chamber pictures. In principle the points A and B should be two vertices of an equilateral triangle having a common center point with the triangle I, II, III and with its sides parallel to the latter. The orientation of the two triangles is not perfect in practice, however, and in analyzing bubble chamber pictures account must be taken of this fact. This is done by specifying the components (a_i, b_i) of the vector $\vec{AB}$ in each of the three principal coordinate systems.

Within these coordinate systems for all view arithmetic

$$3 + 1 = 1.$$

Calculations

Frame Processing - For each frame to be processed the computer calculates some geometric transformation factors. The input data (from magnetic tape) are the quantities

$A_X^i, A_Y^i, B_X^i, B_Y^i, X(P_{jk}^i), Y(P_{jk}^i)$ where

$X(P_{jk}^i)$ is the X coordinate of the k^{th} point of the j^{th} event in the i^{th} view and P denotes a point ($k = 1$ for an E event; $k = 2$ for a P event and $k \geq 3$ for a C event).

We develop the scaling factors for each view as follows:

$$(\sigma^i)^2 = (B_X^i - A_X^i)^2 + (B_Y^i - A_Y^i)^2$$

then

$$A^i = \frac{B_X^i - A_X^i}{(\sigma^i)^2} \quad \text{and} \quad B^i = \frac{B_Y^i - A_Y^i}{(\sigma^i)^2}$$

yielding the rotation coefficients for each frame

$$(T^i)^{11} = a_i A^i + b_i B^i$$

$$(T^i)^{12} = a_i B^i + b_i A^i$$

Event Processing - The first step in the event calculations is a translation of table coordinates of each event point $X(P_{jk}^i), Y(P_{jk}^i)$ to a coordinate system with the fiducial mark A as the origin. (The reader should note that fiducial quantities having upper case X and Y subscripts denote table coordinates and those with lower case x and y subscripts are input constants).

$$X_A(P_{jk}^i) = X(P_{jk}^i) - A_X^i$$

$$Y_A(P_{jk}^i) = Y(P_{jk}^i) - A_Y^i$$

Then calculate an "uncorrected" real space coordinate for each point as follows:

$$\hat{x}(P_{jk}^i) = A_x^i + (T^i)^{11} X_A(P_{jk}^i) + (T^i)^{12} Y_A(P_{jk}^i)$$

$$\hat{y}(P_{jk}^i) = A_y^i - (T^i)^{12} X_A(P_{jk}^i) + (T^i)^{11} Y_A(P_{jk}^i)$$

The correction for the index of refraction of the glass and contents of the bubble chamber is now done as follows:

$$V(P_{jk}^i) = (\hat{x}(P_{jk}^i))^2 + (\hat{y}(P_{jk}^i))^2$$

$$k(P_{jk}^i) = 1 + hV(P_{jk}^i)$$

yielding the corrected coordinates

$$x(P_{jk}^i) = k(P_{jk}^i) \hat{x}(P_{jk}^i)$$

$$y(P_{jk}^i) = k(P_{jk}^i) \hat{y}(P_{jk}^i)$$

The rotated coordinates of each point are calculated as

$$\hat{x}_R(P_{jk}^i) = \alpha_i \hat{x}(P_{jk}^i) + \beta_i \hat{y}(P_{jk}^i)$$

$$\hat{y}_R(P_{jk}^i) = \beta_i \hat{x}(P_{jk}^i) - \alpha_i \hat{y}(P_{jk}^i)$$

then corrected

$$x_R(P_{jk}^i) = k(P_{jk}^i) \hat{x}_R(P_{jk}^i)$$

$$y_R(P_{jk}^i) = k(P_{jk}^i) \hat{y}_R(P_{jk}^i)$$

Event Generation

The event table is generated after each point's table coordinates have been transformed to a set of 12 space coordinates. The event matching procedure is based on the condition that corresponding points in each of the three views will have equal y or y_R coordinates.

All events may not be measured in all views. We consider the views i_1, i_2, i_3 so the number of events in view i_j is greater than in view i_{j+1} for $j = 1, 2$. Matching is performed first with views i_1 and i_2 , the criteria for a match being

$$(y_R(P_{jk}^{(i)}) - y_{jk}^{(i)})^2 < \delta. \quad (1)$$

When the necessary matching attempts have been performed, the procedure goes to i_1 and i_3 , the criteria being

$$(y(P_{jk}^{(i)}) - y_R(P_{jk}^{(i)}))^2 < \delta. \quad (2)$$

After all possible matching attempts for views i_1 and i_3 have been made, views i_2 and i_3 are checked for the existence of any unpaired events. If such events exist, the view i_3 will be designated as i_1 and the matching procedure will be performed on these unpaired events remaining in the two views as per equation (1).

It should be noted that only the first point of a C track will be used in the matching procedure but both points of a P track are matched. In other words, when a match is found for the first points of a P track, the second points will then be matched. If the second points do not match, the event will be treated as not matching. However, if matching of the first points fails, the points are reversed and the matching procedure is repeated. If the first points of a C track match, the C events in the two views being considered will then be checked to determine if they are functional. If the events are good, a curvature check is made to ascertain whether the events match. If the two curvatures are opposite in sign, the events will be treated as matching, otherwise they do not match.

Calculation of the Three Space Coordinates

The method of calculation of the three space coordinates of a point is a function of the number of views in which a corresponding match was found. For each event the three space coordinates and the quantity Q are calculated only for the singular points of that event.

$$\gamma(P_{jk}^i) = \frac{d_i}{x_R(P_{jk}^{i-1}) + x(P_{jk}^i)}$$

$$\tilde{X}(P_{jk}^i) = \gamma(P_{jk}^i) x(P_{jk}^i)$$

$$\tilde{Y}(P_{jk}^i) = 1/2(y(P_{jk}^i) + y_R(P_{jk}^{i-1}))$$

$$\tilde{Z}(P_{jk}^i) = \ell(\gamma(P_{jk}^i) - 1)$$

$$\tilde{Q}(P_{jk}^i) = (y_R(P_{jk}^i) - y(P_{jk}^{i+1}))^2$$

For views 1 and 2 only, calculate

$$X(P_{jk}^{(2)}) = \tilde{Y}(P_{jk}^{(2)})\beta_1 + [d_2 - \tilde{X}(P_{jk}^{(2)})]\alpha_1$$

$$Y(P_{jk}^{(2)}) = [d_2 - \tilde{X}(P_{jk}^{(2)})]\beta_1 - \tilde{Y}(P_{jk}^{(2)})\alpha_1$$

$$Z(P_{jk}^{(2)}) = \tilde{Z}(P_{jk}^{(2)})$$

$$Q(P_{jk}^{(2)}) = \tilde{Q}(P_{jk}^{(2)})$$

For views 1 and 3 only, calculate

$$X(P_{jk}^{(1)}) = \tilde{X}(P_{jk}^{(1)})$$

$$Y(P_{jk}^{(1)}) = \tilde{Y}(P_{jk}^{(1)})$$

$$Z(P_{jk}^{(1)}) = \tilde{Z}(P_{jk}^{(1)})$$

$$Q(P_{jk}^{(1)}) = \tilde{Q}(P_{jk}^{(1)})$$

II A-10

For views 2 and 3 only, calculate

$$X(P_{jk}^{(3)}) = d_1 - \widetilde{X}(P_{jk}^{(3)})\alpha_3 - \widetilde{Y}(P_{jk}^{(3)})\beta_3$$

$$Y(P_{jk}^{(3)}) = \widetilde{X}(P_{jk}^{(3)})\beta_3 - \widetilde{Y}(P_{jk}^{(3)})\alpha_3$$

$$Z(P_{jk}^{(3)}) = \widetilde{Z}(P_{jk}^{(3)})$$

$$Q(P_{jk}^{(3)}) = \widetilde{Q}(P_{jk}^{(3)})$$

If only one view is present

$$X(P_{jk}) = Y(P_{jk}) = Z(P_{jk}) = Q(P_{jk}) = 0$$

If all three views are present we calculate the average coordinates

$$X(P_{jk}) = \frac{X(P_{jk}^{(1)}) + X(P_{jk}^{(2)}) + X(P_{jk}^{(3)})}{3}$$

$$Y(P_{jk}) = \frac{Y(P_{jk}^{(1)}) + Y(P_{jk}^{(2)}) + Y(P_{jk}^{(3)})}{3}$$

$$Z(P_{jk}) = \frac{Z(P_{jk}^{(1)}) + Z(P_{jk}^{(2)}) + Z(P_{jk}^{(3)})}{3}$$

$$Q(P_{jk}) = \frac{Q(P_{jk}^{(1)}) + Q(P_{jk}^{(2)}) + Q(P_{jk}^{(3)})}{3}$$

When the three space coordinates have been obtained we can easily compute the neutron and proton lengths and energies, the radius and depth. All computed quantities are printed but magnetic tape output is written only when there is exactly one curved track in a frame, that track has all three views and the proton and neutron energies, the radius, depth and Q value do not exceed limits set by the program. The tape output from this program is then used as input data to the next program in the Argonne High Energy Physics sequence which completes the studies of the capture process.

COORDINATE SYSTEMS

FIG. I

HIGH-ENERGY INTER-NUCLEAR CASCADES*

A. Van Ginneken et. al.Abstract - American Physical Society Meeting, Minneapolis, June 20-22, 1966

The development of inter-nuclear cascades induced by high-energy (0.1 - 3.0 Gev) particles in large targets of complex nuclei has been studied by a Monte-Carlo calculation¹ using Maniac III and by radiochemical experiments. The particles considered in the calculation were fast nucleons and charged pions emitted in the inelastic particle-nucleus interactions. Energy loss by ionization of charged particles and the absorption of stopped negative pions were included. The basic input data used at present are the results of the intra-nuclear cascade Monte-Carlo calculations of Metropolis, et. al.² The calculation yields the fluxes of the particles as a function of energy and position in the target. These can be used to calculate such quantities as rates of specific nuclear transformations. The production of Zn^{65} and Mn^{54} in Cu foils, embedded in large Fe and Cu targets, has been calculated and is compared with experimental data obtained from proton irradiations at 0.45, 1.0, and 3.0 Gev.

*Work supported in part by the U. S. Atomic Energy Commission and the National Aeronautics and Space Administration.

¹ See also: W. E. Kinney, ORNL-3610, 1964; J. C. D. Milton and J. S. Fraser, Bull. Am. Phys. Soc. 9, 445, 1964.

² N. Metropolis, et. al., Phys. Rev. 110, 185 (1958); N. Metropolis et. al., Phys. Rev. 110, 204 (1958).

ON THE CONVERGENCE OF AN ACCELERATED ITERATIVE METHOD IN THE SOLUTION OF SINGULAR EQUATIONS

W. V. Petryshyn

1. Introduction. Let H be a Hilbert space and let T be a linear selfadjoint operator in H such that T is a contraction (i.e., $\|Tx\| \leq \|x\|$, $x \in H$) and $\lambda = -1$ is not an eigenvalue of T . In [5] Krasnoselsky proved that if, for a given f in H , the equation

$$(1) \quad x = Tx + f, \quad f \in H,$$

is solvable (possibly not uniquely), then the sequence $\{x_n\}$ determined by

$$(2) \quad x_n = Tx_{n-1} + f, \quad n = 1, 2, 3, \dots,$$

converges to a solution of (1) for any x_0 in H . More complete results concerning the method (2) with $T = I - \tau A$ were obtained independently by Bialy [2], who used it for the solution of Eq. (i): $Ax = g$ with $A \geq 0$. In their analysis of the convergence of (2) both authors employed the rather sophisticated spectral decomposition theory for selfadjoint operators. The convergence of (2) for the solution of Eq. (i), where T is an iteration operator associated with the go-method, was also investigated by the author [7]. In case T is a Fredholm integral operator of the first kind, the problem was treated by Fridman [3].

It is known that very often the ordinary iterative method (2) converges rather slowly and, therefore, it is sometimes necessary to increase somehow its speed of convergence. The purpose of the present note is to present an accelerated iterative method for the solution of Eq. (1) and to show that, for T satisfying the conditions imposed by Krasnoselsky, the method converges if and only if Eq. (1) is solvable. It should be added that our proof is of quite an elementary character. When the problem is to solve a linear equation of the form

$$(3) \quad Bx = g, \quad g \in H,$$

with a nonselfadjoint bounded operator B , then, if (3) is solvable, it can always be reduced to an equivalent equation of the form (1) with the corresponding operator T satisfying all the above conditions. Thus in case Eq. 3 is solvable the corresponding accelerated method always converges to a solution of (3) without any further assumption. In fact, we shall obtain a more complete result concerning Eq. (3) and the convergence of the corresponding accelerated method.

2. Accelerated iterative method. Let $N = N(I-T)$ and $R = R(I-T)$ denote respectively the null and the range space of $(I-T)$; let P denote the orthogonal projection of H onto $N = H \ominus \overline{R}$, where $\overline{R}$ is the closure of R . Our first result is the following:

Theorem 1. If T is a selfadjoint contraction in H such that $\lambda = -1$ is not an eigenvalue of T , then for any x_0 in H the sequence $\{y_k\}$ determined by the accelerated iterative method

$$(4) \quad y_0 = x_0, \quad y_1 = T^2 x_0 + (I+T)f, \quad y_k = T^{2^{k-1}} y_{k-1} + U_{2^{k-1}} f, \quad k = 2, 3, \dots,$$

where $A^{2^{k-1}}$ and $U_{2^{k-1}}$ are determined by the recurrence formulas

$$(5) \quad U_{2^{k-1}} = (I + T^{2^{k-2}}) U_{2^{k-2}}, \quad U_1 = I; \quad T^{2^{k-1}} = (T^{2^{k-2}})^2, \quad k = 2, 3, \dots,$$

converges to a solution of (1) if and only if (1) is solvable. In the latter case $y_k \rightarrow x^* + Px_0$, where x^* is the minimal solution of (1) which belongs to $\overline{R}$.

Proof. The proof uses the following lemma which is interesting in its own right and which is proved by the arguments similar to those used in [1, 4].

Lemma 1. If T satisfies the conditions of Theorem 1, then

$$(6) \quad T^{2^m} \rightarrow P$$

as $m \rightarrow \infty$, where (6) means that $T^{2^m} x \rightarrow Px$ for each x in H .

Proof of Lemma 1. It is easy to see that if we put $B = T^2$, then $B^m = T^{2^m}$, $m = 1, 2, 3, \dots$, $0 \leq B \leq 1$, and for every x in H

$$(7) \quad (Bx, x) \leq (x, x), \quad (Bx, Bx) \leq (Bx, x).$$

Furthermore, since $\lambda = -1$ is not an eigenvalue of T , $N(I-B) = N(I-T)$.

Let x be an arbitrary fixed element in H and let $\alpha_m = (B^m x, x)$. Then, in view of (7), $\{\alpha_m\}$ is a monotonically decreasing sequence of non-negative numbers and, therefore, has a limit: $\lim_m \alpha_m = \alpha \geq 0$. This implies that

$$\lim_{n,m} \left\{ \|B^m x - B^n x\|^2 \right\} = \lim_{n,m} \left\{ \alpha_{2m} - 2\alpha_{n+m} + \alpha_{2n} \right\} = 0.$$

Hence there exists a unique element $\tilde{x}$ in H such that $\lim_m B^m x = \tilde{x}$.

If we define the operator E on H by $Ex = \tilde{x}$, then E is clearly linear and, moreover, $E = P$. Indeed, since $T^{2m} x = B^m x \rightarrow Ex$, the passage to the limit in

$$(B^{2m} x, y) = (B^m x, B^m y) = (x, B^{2m} y), \quad x, y \in H,$$

implies that $E = E^*$ and $E^2 = E$, i.e., E is a projection. Furthermore, $R(E) = N(I-T)$. In fact, if $x \in N(I-T)$, then $x \in N(I-B)$ and

$B^m x = x$ for all m . Hence $Ex = x$, i.e., $x \in R(E)$. On the other hand, if $x \in R(E)$, then $Ex = x$ and $\lim_m B^{m+1} x = Ex = \lim_m B(B^m x) = BEx$. Since $Ex = x$, the last equality shows that $Bx = x$, i.e., $x \in N(I-B) = N(I-T)$ and $E = P$.

Sufficiency part of Theorem 1. Suppose that, for a given f in H ,

Eq. (1) is solvable. Then there exists a unique solution x^* of (1) of

smallest norm which belongs to $\overline{R}$. To prove the convergence of y_k to the general solution $x^* + Px_0$ of (1), we first note that (4)-(5) is equivalent to

$$(8) \quad y_k = T^{2^k} x_0 + U_{2^k} f, \quad y_0 = x_0, \quad k = 1, 2, 3, \dots$$

Since for every k the solution x^* satisfies also the equation

$$(9) \quad x^* = T^{2^k} x^* + U_{2^k} f$$

we see that, by subtracting (9) from (8), the error $(y_k - x^*)$ satisfies the relation

$$(10) \quad y_k - x^* = T^{2^k} (x_0 - x^*).$$

Because $\{T^{2^k}\}$ forms a subsequence of $\{T^{2^m}\}$ and, by Lemma 1, the latter converges strongly to P , it follows from (10) and the fact that x^* lies in $\overline{R}$ that

$$y_k \rightarrow x^* + P(x_0 - x^*) = x^* + Px_0.$$

Necessity part of Theorem 1. Let $N^\perp$ and $P^\perp$ denote respectively the orthogonal complement of $N = N(I-T)$ in H and the projection of H onto $N^\perp$. Then, as is well known,

$P^\perp = I - P$, $N^\perp \supset R$, every element x in H has a unique representation $x = Px + P^\perp x$, and $P^\perp(I - T) = (I - T)P^\perp$. This implies that $PT = TP = P$ and, consequently, that $PT^r = T^r P = P$ for every integer $r > 0$. This, the definition of U_{2^k} , and (8) imply that

$$(11) \quad Py_k = Px_0 + 2^k Pf.$$

Suppose now that Eq. (1) has no solution. Then $f \notin R$. Thus, if $Pf \neq 0$, then (11) shows that y_k certainly diverges, while if $Pf = 0$, then the assumption that y_k converges to some element y in H leads to a contradiction. To see this note first that the equality

$$U_{2^k} f = (I + T + T^2 + \dots + T^{2^k - 1})f$$

can also be written in the equivalent form

$$(12) \quad U_{2^k} f = TU_{2^k} f + f - T^{2^k} f$$

Hence, in view of (12), (8) can be put into the form

$$(13) \quad y_k = T^{2^k} x_0 + TU_{2^k} f + f - T^{2^k} f$$

Since $T^{2^k} x_0 \rightarrow Px_0$ and, by assumption, $y_k \rightarrow y$ the equality (8) implies that $U_2^k f \rightarrow y - Px_0$. Thus, using the fact that T is continuous, the passage to the limit in (13) yields the equality

$$y = Px_0 + T(y - Px_0) + f \in Pf.$$

Since $TP = P$ and $Pf = 0$, the last equality shows that y is a solution of (1). This contradicts the condition that $f \notin R$ and thus completes the proof of Theorem 1.

Our second result deals with the iterative solution of Eq. (3).

Theorem 2. Let $\tilde{T} = I - \alpha B^*B$, where α is a real parameter such that

$$(14) \quad 0 < \alpha < \frac{2}{\|B^*B\|},$$

and let π denote the orthogonal projection of H onto $\overline{R(B)}$. For any given x_0 in H determine the sequence $\{u_k\}$ by the process

$$(15) \quad u_0 = x_0, \quad u_1 = \tilde{T}^2 x_0 + (I + \tilde{T})f, \quad u_k = \tilde{T}^{2^{k-1}} u_{k-1} + \tilde{U}_{2^{k-1}} f, \quad k = 2, 3, \dots,$$

where $f = \alpha B^*g$ and $\tilde{T}^{2^{k-1}}$ and $\tilde{U}_{2^{k-1}}$ are determined by the formulas analogous to (5). Then $\{u_k\}$ converges to a solution of

$$(16) \quad Bx = \pi g$$

if and only if Eq. (16) is solvable. In the latter case $u_k \rightarrow u^* + \pi^\perp x_0$,
where u^* is the minimal solution of (16) which belongs to $\overline{R(B^*)}$.

Proof of Theorem 2. Note first that since $H = N(B^*) \oplus \overline{R(B)}$
 and $\pi H = \overline{R(B)}$, every solution of (16) is also a solution of

$$(17) \quad B^* Bx = B^* x$$

and, conversely, i.e., (16) and (17) are equivalent. Indeed, in virtue
 of the equalities $g = \pi g + \pi^\perp g$ and $B^* g = B^* \pi g$, we see that if y is
 a solution of (16), then it is also a solution of (17); on the other hand, it
 also follows that every solution of y of (17) is also a solution of $B^* By =$
 $B^* \pi g$. Hence, $(By - \pi g)$ belongs simultaneously to $N(B^*)$ and to $\overline{R(B)}$
 and, therefore, $By - \pi g = 0$. Thus the equivalence of (16) and (17) is
 proved. Furthermore, it is obvious that for any $\alpha \neq 0$, Eq. (17) is
 itself equivalent to the equation

$$(18) \quad x = (I - \alpha B^* B)x - \alpha B^* g.$$

It is easy to see that if we put $\tilde{T} = (I - \alpha B^* B)$ and $\tilde{f} = -\alpha B^* g$,
 then for α satisfying the condition (14) the operator $\tilde{T}$ satisfies all the
 conditions of Theorem 1. Furthermore, equation (18) which is equivalent
 to (16) has the form (1) and the process (15) is precisely the accelerated
 method (4). Finally, since $\overline{R(I - \tilde{T})} = \overline{R(B^* B)} = \overline{R(B^*)}$, the validity of Theo-
 rem 2 follows from Theorem 1.

Remarks. 1. Since $\Pi g = g$ whenever Eq. (3) is solvable, Theorem 2 implies that in this case the accelerated method (15) always converges to a general solution of (3).

2. Instead of using the method (4)-(5) it may prove on occasions to be more useful or convenient to use the equivalent accelerated method given by (8) and (5). For a convergent $n \times n$ matrix T the latter method was suggested by Naishul [6]. It is easy to see that k steps with the accelerated method (4)-(5) are equivalent to $n = 2^k$ steps with the ordinary iterative process (2).

3. Lemma 1 suggests the consideration of another iterative method for the solution of Eq. (1) defined by

$$(19) \quad x_k = T^2 x_{k-1} + (I+T)f, \quad k = 1, 2, 3, \dots,$$

where x_0 is an arbitrary initial approximation. It is obvious from the above discussion that the assertions of Theorem 1 and of Theorem 2 are also valid for the method (19) and that k steps with the method (19) are equivalent to $2k$ steps with the method (2).

4. If in Eq. (3) the operator B is selfadjoint and non-negative, then the passage to Eq. (17) is unnecessary. In this case $\tilde{T}$ and $\tilde{f}$ in (15) are given by $\tilde{T} = I - \alpha B$ and $\tilde{f} = -\alpha g$ with α satisfying the condition $0 < \alpha < \frac{2}{\|B\|}$. It should be added, however, that the condition $B \geq 0$ cannot be replaced by the mere requirement that B be selfadjoint since in the latter case we, in general, would not be

able to find a parameter α such that $T = I - \alpha B$ would satisfy the conditions of Theorem 1.

REFERENCES

1. I. Babuska, Über Schwarzsche Algorithmen in partiellen Differentialgleichungen der mathematischen Physik, Czechoslovak Math. J. 8 (1958), 328-348.
2. H. Bialy, Iterative Behandlung linearer Funktionalgleichungen, Archive Rat. Mech. Anal. 4 (1959), 166-176.
3. V.M. Fridman, A method for successive approximations for Fredholm integral equations of the first kind, Uspehi, Mat. Nauk 11 (1956), 233-234.
4. S. Hildebrandt, Einige konstruktive Methoden bei Randwertaufgaben für lineare partielle Differentialgleichungssysteme und in der Theorie harmonischer Differentialformen I, 213 (1963), 66-88.
5. M.A. Krasnoselsky, On the solution of equations with selfadjoint operators by the method of successive approximations, Uspehi Mat. Nauk, 15 (1960), 161-165.
6. A.B. Naishul, Improving the convergence of methods of successive approximation for linear equations, Dokl. Akad. Nauk SSSR 158 (1964), 279-280.
7. W.V. Petryshyn, The generalized overrelaxation method for the approximate solution of operator equations in Hilbert space, J. Soc. Indust. Appl. Math. 10 (1962), 675-690.

FURTHER REMARKS ON NONLINEAR P-COMPACT OPERATORS IN BANACH SPACE

W. V. Petryshyn

1. Introduction. In recent papers [8, 9] the author, using rather elementary arguments, derived a number of results¹ concerning the existence and the construction of solutions and of fixed points for a class of nonlinear equations involving bounded projectionally-compact¹ (P-compact) operators defined on a real Banach or Hilbert space with a basis. These results extended and simplified similar results for quasicompact operators obtained by Kaniel [5]. At the same time we deduced from our theorems certain basic results on bounded monotone operators obtained previously by Minty [7], Browder [1, 2] and others.

The purpose of this paper is twofold: First, we extend our main results derived in [8, 9] to P-compact operators without assuming their boundedness; second, using our extended results, we generalize to our class of operators the recent results¹ of Granas [3, 4] concerning the solvability of nonlinear equations and the geometrical intersection theorem involving quasibounded¹ completely continuous operators. At the same time we obtain some new results for P-compact and monotone operators in Hilbert space.

2. Extended results for P-compact operators. Let X be a real Banach space with the property that there exists a sequence $\{X_n\}$ of

¹ For the precise definitions of the concepts and the statements of the results mentioned in the Introduction, see the succeeding sections of this paper.

finite dimensional subspaces X_n of X , a sequence of linear projections

$\{P_n\}$ defined on X , and a constant $K > 0$ such that

$$(1) \quad P_n X = X_n, \quad X_n \subset X_{n+1}, \quad n = 1, 2, 3, \dots, \quad \overline{\bigcup_n X_n} = X,$$

$$(2) \quad \|P_n\| \leq K, \quad n = 1, 2, 3, \dots$$

Let B_r denote the closed ball in X of radius $r > 0$ about the origin and let S_r denote the boundary of B_r . Let the symbols " $\rightarrow$ " and " $\rightharpoonup$ " denote the strong and weak convergence in X , respectively. In what follows we consider the class of P-compact operators which were studied by the author in [8, 9] and which are defined as follows:

Definition 1. A nonlinear operator A mapping X into itself is called projectionally-compact (P-compact) if $P_n A$ is continuous in X_n for all sufficiently large n and if for any constant $p > 0$ and any bounded sequence $\{x_n\}$ with $x_n \in X_n$ the strong convergence of the sequence $\{g_n\} \equiv \{P_n A x_n - p x_n\}$ implies the existence of a strongly convergent subsequence $\{x_{n_i}\}$ and an element x in X such that $x_{n_i} \rightarrow x$ and $P_{n_i} A x_{n_i} \rightarrow A x$.

The results of this paper will be based essentially on the following theorem which for bounded P-compact operators was proved by the author in [8, 9].

Theorem 1. Suppose that A is P-compact. Suppose further that for given $r > 0$ and $\mu > 0$ the operator A satisfies both of the following

conditions: (Δ) There exists a number $c(r) > 0$ such that if, for any n , $P_n Ax = \lambda x$ holds for x in S_r with $\lambda > 0$, then $\lambda \leq c(r)$.

(π_μ) If for some x in S_r the equation $Ax = \alpha x$ holds then $\alpha < \mu$. Then there exists at least one element u in $(B_r - S_r)$ such that

$$(3) \quad Au - \mu u = 0.$$

Proof. The proof Theorem 1 follows the same line of argument as the proof of the corresponding Theorem 2 in [9]. As in [9], all we need is to show that the present conditions (Δ) - (π_μ) imply the validity of the following lemma.

Lemma 1. If A satisfies the conditions of Theorem 1, then there exists an integer $n_0 > 0$ such that if $n \geq n_0$ and $P_n Ax = \beta x$ for some x in $S_r \cap X_n$ then $\beta < \mu$.

Proof of Lemma 1. If the assertion of Lemma 1 were not true for any n_0 , we could find a sequence $\{x_n\}$ with $x_n \in X_n \cap S_r$ and a sequence of numbers $\{\beta_n\}$ such that

$$(4) \quad P_n Ax_n = \beta_n x_n, \quad (\beta_n \geq \mu).$$

Hence our condition (Δ) implies that

$$\|P_n Ax_n\| = \beta_n r \leq c(r)r,$$

i. e., $\beta_n \in [\mu, c(r)]$ for each n . Passing to a subsequence, we may assume that $\beta_n \rightarrow \beta$ and $\beta \in [\mu, c(r)]$. This and (4) show that

$$(5) \quad P_n A x_n - \beta x_n = (\beta_n - \beta) x_n \rightarrow 0, \quad (n \rightarrow \infty).$$

Since A is P -compact, (5) implies the existence of a strongly convergent subsequence, which we again denote by $\{x_n\}$, and an element x in $S_r \cap X$ such that $x_n \rightarrow x$ and $P_n A x_n \rightarrow Ax$. This and (5) imply that $Ax - \beta x = 0$ for x in S_r and $\beta \geq \mu$, in contradiction to condition (π_μ) of Theorem 1.

The proof of Theorem 1 is then completed in exactly the same way as in [9].

Remark 1. It is obvious that if in Definition 1 we require $p < 0$ instead of $p > 0$, then we obtain a theorem analogous to Theorem 1. We need only consider $-A$ instead of A and assume that for given $r > 0$ and $\mu < 0$ instead of conditions (Δ) and (π_μ) the operator A satisfies the conditions:

(Δ^-) There exists a number $c(r) > 0$ such that if, for any n , $P_n A x = \lambda x$ holds for x in S_r with $\lambda < 0$, then $-c(r) \leq \lambda$.

(π_μ^-) If for some x in S_r the equation $Ax = \alpha x$ holds then $\mu < \alpha$.

Remark 2. Let us remark that condition (Δ) is in no way a condition on the size of Ax or even on the size of $P_n A x$. All it says is that when for any x in S_r and any n the vector $P_n A x$ is in the same

direction as x then the $P_n Ax$ are uniformly bounded.

Remark 3. The assertion of Theorem 1 remains valid if condition (Δ) is replaced, for example, by any one of the following stronger conditions whose degree of generality increases in the given order:

(a) A is bounded, i. e., A maps bounded sets in X into bounded sets.

(b) For any given $r > 0$ the set $A(S_r)$ is bounded.

(c) X is a Hilbert space H and, for any given $r > 0$, $(Ax, x) \leq c \|x\|^2$ for all x in S_r and some $c > 0$.

Corollary 1. If A is P -compact and for some $r > 0$ the conditions (Δ) and (π_1) are satisfied on S_r , then A has a fixed point in $(B_r - S_r)$.

Theorem 2. Suppose that A is P -compact. Suppose further that there exists a sequence of spheres $\{S_{r_p}\}$ with $r_p \rightarrow \infty$, as $p \rightarrow \infty$, and two sequences of positive numbers $c_p = c(r_p)$ and $k_p = k(r_p)$ with $k_p \rightarrow \infty$, as $r_p \rightarrow \infty$, such that the following conditions hold:

(Δ_f) Whenever for any given f in B_{k_p} and any n the equation $P_n Ax - \lambda x = P_n f$ holds for x in S_{r_p} with $\lambda > 0$, then $\lambda \leq c_p$.

(π_p) $\|Ax - \mu x\| \geq k_p$ for any $\mu \geq \mu > 0$ and any x in S_{r_p} .

Then for every f in X there exists an element u in X such that

$$(6) \quad Au - \mu u = f.$$

Proof. For any given f in X choose r_p so large that $\|f\| < k_p$.

If we define $Cx \equiv Ax - f$, then obviously C is P -compact. Furthermore, on S_{r_p} the operator C satisfies conditions (Δ) and (π_μ) of Theorem 1.

Indeed, if $P_n Cx = \lambda x$ for any x in S_{r_p} with $\lambda > 0$, then, by the definition of C , $f \in B_{k_p}$ and $P_n Ax - \lambda x = P_n f$. Hence, by (Δ_f) , $\lambda \leq c_p$, i. e., (Δ) is fulfilled. Suppose now that $Cx = \alpha x$ for some x in S_{r_p} . Then $\alpha < \mu$ since, in virtue of (π_p) , the assumption that $\alpha \geq \mu$ would lead to the contradiction

$$\|Cx - \alpha x\| = \|Ax - \alpha x - f\| \geq \|Ax - \alpha x\| - \|f\| \geq k_p - \|f\| > 0.$$

Hence, by Theorem 1, there exists an element u in $(B_{r_p} - S_{r_p})$ such that $Cu = \mu u$, i. e., u is a solution of Eq. (6).

In case x is a Hilbert space we have the following interesting theorem.

Theorem 3. If A is a P -compact mapping of H into itself such that

$$(7) \quad (Ax, x) \leq (A(0), x), \quad x \in H,$$

then for any given $\mu > 0$ the operator $(A - \mu I)$ is onto.

Proof. Let $\{S_{r_p}\}$ be a sequence of spheres in H with $r_p = p + \frac{\|A(0)\|}{\mu}$ and let $\{c_p\}$ and $\{k_p\}$ be two sequences given by $c_p = \mu$ and $k_p = \mu p = \mu r_p - \|A(0)\|$. For any given f in H choose r_p so large that $\|f\| < k_p$. If we put $Cx = Ax - f$, then (7) implies that for all x in H

$$(8) \quad (Cx, x) \leq (C(0), x).$$

It is not hard to show that, in virtue of (8), the above defined sequences

$\{c_p\}$ and $\{k_p\}$ satisfy conditions (Δ_f) and (π_p) of Theorem 2.

In fact, suppose that for any f in B_k and any n the equation

$P_n Ax - \lambda x = P_n f$ holds for x in $S_{r_p}^{k_p}$ with $\lambda > 0$. Then, by our definition of C , $P_n Cx = \lambda x$ and $x \in H_n \cap S_{r_p}^{k_p}$. This and (8) imply that

$$\lambda(x, x) = (P_n Cx, x) = (cx, x) \leq (C(0), x) \leq \|C(0)\| \|x\| \leq (\|A(0)\| + \|f\|) \|x\|$$

from which we derive the inequality $\lambda \|x\| \leq \|A(0)\| + k_p = \mu r_p$. Thus,

$\lambda \leq c_p$, i. e., condition (Δ_f) is satisfied. To verify condition (π_p) note that if for $\eta \geq \mu$ we put $Qx = Ax - \eta x$, then by (7)

$$(Q(0) - Qx, x) = (A(0) - Ax + \eta x, x) = (A(0), x) - (Ax, x) + \eta(x, x) \geq \eta(x, x)$$

Hence $\|Qx - Q(0)\| \geq \eta \|x\| \geq \mu \|x\|$ for each x in H and any $\eta \geq \mu$.

Thus for any x in $S_{r_p}^{k_p}$ and $\eta \geq \mu$

$$\begin{aligned} \|Ax - \eta x\| &= \|Qx\| \geq \|Qx - Q(0)\| - \|Q(0)\| \geq \mu \|x\| - \|A(0)\| = \mu r_p - \|A(0)\| \\ &= \mu p = k_p \end{aligned}$$

which is precisely condition (π_p) . Consequently Theorem 3 follows from Theorem 2.

Corollary 2. If A is P -compact and monotone decreasing, i. e.,

$$(9) \quad (Ax - Ay, x - y) \leq 0, \quad x, y \in H,$$

then for any given $\mu > 0$ the mapping $P = A - \mu I$ is one-to-one and onto.

Proof. The onto part of the assertion of Corollary 2 follows from Theorem 3 since (9) clearly implies (7) while the one-to-one part follows from the fact that $\|Px - Py\| \geq \mu\|x - y\|$ for any x and y in H and any $\mu > 0$.

3. Applications to quasibounded mappings. Let Y and Z be any two real Banach spaces. Following Granas [3] we say that a nonlinear mapping A of Y into Z is quasibounded if there exist two constants $M > 0$ and $q_0 > 0$ such that

$$(10) \quad \|Ax\| \geq M\|x\| \quad \text{for all } x \text{ in } Y \text{ with } \|x\| \geq q_0.$$

If A is quasibounded, then the number $|A|$ defined by

$$(11) \quad |A| = \inf_{q_0 \leq q < \infty} \sup_{\|x\| \geq q} \left\{ \frac{\|Ax\|}{\|x\|} \right\}$$

is called the quasinorm of A . It follows that every bounded linear operator is quasibounded and that its norm coincides with its quasinorm. Furthermore,

as was observed by Granas, every nonlinear mapping of Y into Z which is asymptotically differentiable in the sense of Krasnoselsky [6] is quasibounded. In fact, if A is asymptotically differentiable, then there exists a linear operator A' mapping Y into Z , called the asymptotic derivative of A , such that

$$(12) \quad \lim_{\|x\| \rightarrow \infty} \left\{ \frac{\|A(x) - A'x\|}{\|x\|} \right\} = 0 ;$$

hence it follows easily from (12) that A is quasibounded and that $\|A\| \leq \|A'\|$. Let us add that the class of asymptotically differentiable operators was thoroughly studied in [6].

The purpose of this section is to apply our theorems proved in Section 2 to the generalization of results obtained by Granas [3, 4] for completely continuous quasibounded operators by means of topological arguments.

Theorem 4. Suppose that A is a P -compact and quasibounded mapping of X into itself. If $\mu > M$, then $(A - \mu I)$ is onto.

Proof. Let $\{r_p\}$ be a sequence of real numbers such that $r_p \geq q_0$ for all p and such that $r_p \rightarrow \infty$ as $p \rightarrow \infty$. Then, in view of our conditions, for all z in S_{r_p} and any $\eta \geq \mu$,

$$\|Ax - \eta x\| \geq \eta \|x\| - \|Ax\| \geq \mu \|x\| - M \|x\| = (\mu - M) \|x\| = (\mu - M) r_p.$$

Thus condition (π_p) of Theorem 2 is satisfied with $k_p = (\mu - M) r_p$. Now

suppose that for any f in B_{k_p} and any n the equation $P_n Ax - \lambda x = P_n f$ holds for x in S_{r_p} with $\lambda > 0$. Then, by (2) and (10), the latter equation implies that

$$\lambda r_p = \lambda \|x\| = \|P_n(Ax-f)\| \leq K \|Ax-f\| \leq K \{ \|Ax\| + \|f\| \} \leq K \{ Mr_p + k_p \}.$$

Hence, $\lambda \leq \mu K$, i. e., condition (Δ_f) is satisfied with $c_p = \mu K$ for each p . Consequently, Theorem 4 follows from Theorem 2.

Remark 4. It is not hard to see that Theorem 4 remains valid if instead of assuming that $\mu > M$ we assume that $\mu > |A|$.

Corollary 3. Suppose that A is quasibounded and P -compact with $p < 0$. If $\mu > M$, then $(\mu I + A)$ maps x onto itself.

Proof. The conditions of Corollary 3 imply that $\tilde{A} = -A$ is quasibounded and P -compact with $\tilde{p} = -p > 0$. Hence, by Theorem 4, $(\tilde{A} - \mu I)$ or equivalently the operator $(\mu I + A)$ is onto.

Remark 5. When A is completely continuous (i. e., A is continuous and compact) and $\mu = 1$ Corollary 3 was proved by Granas [3] by the application of the topological fixed point theorem of Rothe [10].

An intersection theorem in X . Suppose that X is a direct sum of the subspaces $V \subset X$ and $W \subset X$, i. e., $X = V \oplus W$, and suppose that P_V and P_W denote the projections of X onto V and W , respectively. It is obvious that P_V and P_W are linear and that

$$(13) \quad \|P_V x\| \leq \|P_V\| \|x\|, \quad \|P_W x\| \leq \|P_W\| \|x\|, \quad x \in X.$$

Suppose further that $f(v) \equiv v + F(v)$ maps V into X and that $g(w) \equiv w + G(w)$ maps W into X , where F and G are quasi-bounded and completely continuous nonlinear mappings. Using Rothe's theorem Granas [4] obtained an interesting intersection theorem by proving that if

$$(14) \quad |F| \|P_V\| + |G| \|P_W\| < 1,$$

then the images $f(V)$ and $g(W)$ have a nonempty intersection, i. e.,

$$f(V) \cap g(W) \neq \emptyset.$$

Here we consider the intersection theorem when either F or G is P -compact and when condition (14) is replaced by a much weaker condition. Our result is based on the application of Theorem 1.

Theorem 5. Let G be a nonlinear mapping of W into X such that the operator $G(-P_W)$ is P -compact and such that to a given $r > 0$ there corresponds a number $c(r) > 0$ with the property that for all x in S_r

$$(15) \quad \|G(-P_W x)\| \leq c(r).$$

Let F be a completely continuous nonlinear mapping of V into X and let $f_\mu(v)$ and $g_\mu(w)$ be the mappings defined respectively from V and W to X by $f_\mu(v) = \mu v + Fv$ and $g_\mu(w) = \mu w + Gw$. If for given $r > 0$

and $\mu > 0$ the operators F and G satisfy the condition

(π) If $Fv + \alpha v = Gw + \alpha w$ for some v in V and w in W with $\|v - w\| = r$ then $\alpha < \mu$, then $f_\mu(V) \cap g_\mu(W) \neq \emptyset$.

Proof. Let us define a nonlinear mapping A of X into X by

$$(16) \quad Ax = G(w) - F(v) \text{ with } w = -P_W x \text{ and } v = P_V x, \quad x \in X,$$

and observe that $f_\mu(V) \cap g_\mu(W) \neq \emptyset$ if and only if the equation

$$(17) \quad Ax = \mu x$$

has a solution in X . Indeed, if x is a solution of (17), then x has a unique representation $x = P_V x + P_W x = v - w$ and, in view of (16), (17) implies that $Gw - Fv = \mu(v - w)$ or that $\mu w + Gw = \mu v + Fv$, i. e., $f_\mu(V) \cap g_\mu(W) \neq \emptyset$. On the other hand, if $v \in V$ and $w \in W$ are two elements such that $f_\mu(V) \cap g_\mu(W) \neq \emptyset$, then $\mu(v - w) = Gw - Fv$. Hence, if we put $x = v - w$, (16) implies that x is a solution of (17).

Thus, to prove Theorem 5 it is sufficient, in view of the above observation and Theorem 1, to show that the operator A defined by (16) is P -compact and satisfies conditions (Δ) and (π_μ).

Let us first show that A is P -compact. Now, by our conditions on G and F , $P_n A$ is certainly continuous in X_n for all sufficiently large n . Further, let $\{x_n\}$ be a bounded sequence so that for any $p > 0$

$$(18) g_n = P_n A x_n - p x_n = P_n G(-P_W x_n) - p x_n - P_n F(P_V x_n) \rightarrow g, \quad x_n \in X_n.$$

Since $\{v_n\} = \{P_V x_n\}$ is bounded and F is completely continuous, there exists a subsequence, which we again denote by $\{x_n\}$, such that

$$F(v_n) = F(P_V x_n) \rightarrow v \text{ and } P_n F(v_n) \rightarrow v, \text{ where } v \text{ is some element in } X.$$

This and (18) imply that

$$\tilde{g}_n \equiv P_n G(-P_W x_n) - p x_n = g_n + P_n F(P_V x_n) \rightarrow g + v, \quad (n \rightarrow \infty).$$

Since $G(-P_W)$ is P -compact there exists a subsequence, again denoted by

$\{x_n\}$, such that $x_n \rightarrow x$ and $P_n G(-P_W x_n) \rightarrow G(-P_W x)$. This and the continuity of F imply that

$$P_n A x_n = P_n G(-P_W x_n) - P_n F(P_V x_n) \rightarrow G(-P_W x) - F(P_V x) = A x$$

i. e., A is P -compact.

Suppose now that $Ax = \alpha x$ for some x in S_r . This then means that $Gw + \alpha w = Fv + \alpha w$ with $\|v - w\| = \|P_V x + P_W x\| = r$. Hence our condition (π) implies that $\alpha < \mu$, i. e., A satisfies condition (π_μ) . Finally we see that for any x in S_r condition (15) and the complete continuity of F imply the inequality

$$\|Ax\| \leq \|G(-P_W x)\| + \|F(P_V x)\| \leq c(r) + c, \quad x \in S_r,$$

where $c > 0$ is such that $\|F(P_V x)\| \leq c$ for all x in S_r . Thus the set $A(S_r)$ is bounded and therefore, by Remark 3(b), A satisfies condition $(\bigwedge)$. Hence, by Theorem 1, Eq. (17) has at least one solution in B_r or, equivalently, the intersection $f_\mu(V) \cap g_\mu(W) \neq \emptyset$.

Corollary 4. Suppose that G and F satisfy all conditions of Theorem 5 except that condition (Π) is replaced by the condition

$$(19) \quad \|\mu w + Gw - (\mu v + Fv)\|^2 = \|Fv - Gw\|^2 - \mu^2 \|v - w\|^2 \text{ for } v \in V, W \in W \text{ with } \|v - w\| = r.$$

Then $f_\mu(V) \cap g_\mu(W) \neq \emptyset$.

Proof. We may assume, without loss of generality, that there are no elements v in V and w in W with $\|v - w\| = r$ such that $f_\mu(v) = g_\mu(w)$. Suppose now that for some x in S_r , or equivalently for some v in V and w in W with $\|v - w\| = r$, we have

$$Fv + \alpha v = Gw + \alpha w. \text{ Then } \|Gw - Fv - \mu(v - w)\|^2 = \|\alpha(v - w) - \mu(v - w)\|^2 = (\alpha - \mu)^2 \|v - w\|^2 \text{ and } \|Gw - Fv\|^2 - \mu^2 \|v - w\|^2 = (\alpha^2 - \mu^2) \|v - w\|^2.$$

Hence, by (19), $(\alpha - \mu)^2 \geq (\alpha^2 - \mu^2)$ or $2\mu^2 \geq 2\mu\alpha$. Since $\mu > 0$, our assumption then implies that $\alpha < \mu$ and, consequently, (18) implies condition (Π) . Corollary 4 then follows from Theorem 5.

Remark 6. In case X is a Hilbert space condition (19) is equivalent to the requirement

$$(20) \quad (G(-P_W x) - F(P_V x), x) \leq \mu \|x\|^2, \quad x \in S_r.$$

Corollary 5. Suppose that F and G are completely continuous and quasibounded, i. e., there exist four constants $M_1 > 0$, $M_2 > 0$, $r_1 > 0$ and $r_2 > 0$ such that

$$(21) \quad \|Fv\| \leq M_1 \|v\| \text{ for every } v \text{ in } V \text{ with norm } \|v\| \geq r_1,$$

$$(22) \quad \|Gw\| \leq M_2 \|w\| \text{ for every } w \text{ in } W \text{ with norm } \|w\| \geq r_2.$$

Suppose further that M_1 and M_2 satisfy the inequality

$$(23) \quad M_1 \|P_V\| + M_2 \|P_W\| \leq 1.$$

then $f_1(V) \cap g_1(W) \neq \emptyset$.

Proof. Let us first remark that, as was shown by Granas, the conditions of Corollary 5 imply the existence of a constant $r > 0$ such that

$$(24) \quad \|G(-P_W x) - F(P_V x)\| \leq \|x\| \text{ for every } x \text{ in } X \text{ with } \|x\| \geq r,$$

i. e., the operator $A(x) = G(-P_W x) - F(P_V x)$ is quasibounded. Assuming, without loss of generality, that there are no elements v in V and w in W with $\|v - w\| = r$ such that $f_1(v) = g_1(w)$, it is easy to see that whenever $Fv + \alpha v = Gw + \alpha w$ for some v in V and w in W with $\|v - w\| = r$, then (24) implies that $\alpha < 1$. Hence condition (π) of Theorem 5 holds for $\mu = 1$. Furthermore, since G is completely

continuous; (15) is clearly satisfied and, by Theorem 3 in [9], $G(-P_W)$ is P-compact. Consequently, Corollary 5 follows from Theorem 5.

Remark 7. For the sake of completeness let us show that the conditions of Corollary 5 imply the validity of (24) for some $r > 0$.

First let $r_0 = \max \{r_1, r_2\}$ and let $c > 0$ be a constant such that

$$\|Fv\| \leq c \text{ for } v \text{ in } V \text{ with } \|v\| \leq r_0 \text{ and } \|Gw\| \leq c \text{ for } w \text{ in } W \\ \text{with } \|w\| \leq r_0.$$

Taking $r = \max \left\{ 2r_0, \frac{c}{1 - M_0} \right\}$, where $M_0 = \max \{M_1 \|P_V\|, M_2 \|P_W\|\} < 1$, we obtain (24). Indeed, (24) follows trivially from (21), (22), and (23) if for $x = v - w$ with $\|x\| \geq r$ we have $\|v\| \geq r_0$ and $\|w\| \geq r_0$. On the other hand, if for $\|x\| \geq r$ one of the conditions $\|v\| \geq r_0$ or $\|w\| \geq r_0$ is not satisfied (e. g. $\|w\| < r_0$), then by our definition of c and M_0 we get the desired inequality

$$\|G(-P_W x) - F(P_V x)\| \leq \|G(w)\| + \|F(v)\| \leq c + M_0 \|x\| \leq (1 - M_0) \|x\| \\ + M_0 \|x\| = \|x\|$$

Remark 8. Let us observe finally that the results obtained in Section 5 [9] concerning the applicability of the projection method to the approximate solution of either the equation (3) or (6) remain also valid for unbounded P-compact operators provided, of course, that they satisfy the corresponding conditions assumed in [9].

REFERENCES

1. F. E. Browder, The solvability of nonlinear functional equations, Duke Math. J. 30 (1963), 554-566.
2. _____, Variational boundary value problems for quasi elliptic equations of arbitrary order, Proc. Nat. Acad. Sci. 5 (1965) 31-37.
3. A. Granas, On a class of nonlinear mapping in Banach spaces, Bull. Acad. Polon. Sci. Cl. III, 5 (1957), 867-870.
4. _____, On a geometrical theorem in Banach space, Bull. Acad. Polon. Sci. Cl. III, 5 (1957), 873-877.
5. S. Kaniel, Quasicompact nonlinear operators in Banach space and applications (to appear).
6. Krasnoselsky, Topological methods in the theory of nonlinear integral equations, State Publ. House, Moscow, 1956.
7. G. J. Minty, Monotone (nonlinear) operators in Hilbert space, Duke Math. J. 29 (1962), 341-346.
8. W. V. Petryshyn, On a fixed point theorem for nonlinear P-compact operators in Banach space (to appear in Bull. Amer. Math. Soc.).
9. _____, On nonlinear P-compact operators in Banach space with applications to constructive fixed point theorems (to appear in J. Math. Anal. and Appl.)
10. E. Rothe, Zu der topologischen Ordnung und der Vektorfeldern in Banachschen Raumen, Compositio Math. 5 (1957).

POLARIZATION IN STELLAR DYNAMICAL SYSTEMS

R. H. Miller

ABSTRACT

Differences between two nearly identical stellar dynamical systems are studied; they tend to grow very rapidly. A displacement of a star in one system induces dipolar forces in the difference system; the resulting change of motion of the neighboring stars polarizes the difference medium in such a way as to screen distant stars while leading to an exponentially growing displacement with a characteristic time of $\sim (G_1 \rho)^{-1/2}$. Forces in a stellar dynamical system are balanced so that moving one star causes the surrounding stars to move collectively (polarize the medium) in such a way that their reaction on the original star enhances the initial displacement. Orbits are affected in the sense that for any orbit there exists another orbit, initially arbitrarily close, which departs from it with a separation increasing exponentially with the time. Epicyclic motions and random velocities tend to reduce the growth rate. Some consequences of the polarization are discussed in a preliminary way.

I. INTRODUCTION

A computer investigation of the n-body problem in stellar dynamics (Miller 1964) revealed an irreversible aspect of the motion that was characterized by an exponential process whose characteristic time was interpreted as a kind of relaxation time. It was argued that the effects

were of physical, and not numerical origin. Collective phenomena also emerged. Further computer studies were not pursued because a satisfactory formulation of the questions raised by these observations was not evident.

These questions include the rate of information propagation through the system, the number of stars affected in a large system, and the nature of any cooperative phenomena that may exist. An understanding of these matters is essential to an appreciation of which functionals are affected by the resulting numerical instability.

A perturbation approach to this problem is developed in this paper. It is reminiscent of a simplified stability calculation, and provides a picture of the physical processes involved. The notion of parallel evolution of two nearly identical systems, as described in the paper on the computer study, is central to the development.

II. FORMALISM

Imagine two stellar dynamical systems that are identical in every respect. Each will develop in some self-consistent way that is governed by the fundamental Newtonian equations of motion. If they are identical ab initio, they will forever remain identical. Now imagine a star in one of these two systems being displaced by a tiny amount from its initial position. What will be its subsequent motion? How will it affect the surrounding regions? The tiny displacement should be small compared to the distance to any neighboring star, so the identity of the stars is not confused.

All the forces in the second system (the one with the perturbation) are identical to the corresponding forces in the first system except that there is a force of dipolar character added to the second system. It is produced by the dipole created by moving the perturbed star, and consists of a positive mass at the location of the star in the second system and a negative mass at the position of the corresponding star in the first system. The dipole has a moment $m \delta$, if m be the mass of the star that was moved, and δ be the (vector) displacement (in the sense of $\chi_i^{(2)} - \chi_i^{(1)} = \delta_i$; (1) and (2) refer to the first and second systems and i is a tensor index.)

The potential and force resulting from this dipole may be calculated in the usual way. All the forces acting on a star of the second system may be calculated as the (vector) sum of all the forces in the first system plus the force due to the dipole. Since the sum of all the forces in the first system is balanced by the accelerations in the first system, the forces in the second system will be balanced by the accelerations in the first system except for the force due to the dipole. A set of displacements will develop, one for each star, arising from the force field surrounding the perturbed star. These displacements should also be measured as the difference of coordinates between the stars of the second and the first systems. Each induced displacement itself corresponds to a dipole and produces its own force field. According to the continuum picture, the medium is polarizable in the sense of electrostatics. These induced dipoles influence the original star and screen the surrounding medium.

The reaction of the surrounding stars (or of the medium) on the original star is in such a direction as to augment the displacement; the displacement grows if the reaction overrides any restoring force. Since the forces due to the perturbation are dipolar, integrals over the system may be expected to converge at large distances, and an infinite homogeneous system may be considered.

The study of this effect is undertaken with the help of some simplifying assumptions to keep the problem manageable. Many different formulations are possible; a few illustrations follow. The arguments presented are very simple, to illustrate the ideas. They could be made more precise, but that will not be attempted.

a) Static Approximation

The displacement of the first star gives rise to a dipole of moment $p_i = m \delta_i^{(0)}$, where $\delta_i^{(0)}$ is the displacement in the perturbed system relative to the unperturbed, and m is the mass of the star that was moved. All stars will be taken to have the same mass, although that is not essential to the argument. The force per unit mass at some point x_j (taken about the dipole as origin of coordinates; summation convention is implied with $r^2 = x_j x_j$), is

$$F_i = + G m \frac{\delta_j^{(0)}}{r^3} \left(\delta_{ij} - 3 \frac{x_i x_j}{r^2} \right) = G m \delta_j^{(0)} D_{ij}(\vec{r}). \quad (1)$$

Properly, the forces acting on stars are time-dependent through the

variation in χ_j , but if the continuum model is invoked, there will be at each point of space a distribution of stars, as many with positive as with negative velocity components. The first-order velocity dependence in F_i will cancel (no anisotropy in the velocity field). The force per unit mass produces an acceleration $\ddot{\delta}_i^{(1)}$.

It is convenient to introduce σ as an operator of time-differentiation that may be manipulated algebraically in the usual operational sense (Jeffreys and Jeffreys, 1962); then the equations may be written

$$\sigma^2 \delta_i^{(1)} = G_m \delta_j^{(0)} D_{ij}(\vec{r}). \quad (2)$$

The integration constants may be taken to be zero. With N stars per unit volume, the resulting dipole moment per unit volume is

$$P_i = N m \delta_i^{(1)}. \quad (3)$$

(The susceptibility may be regarded as Nm/σ^2 .)

In an infinite homogeneous isotropic medium, the polarization gives rise to no new multipoles; otherwise higher multipoles would be generated. The "polarization" of the intervening matter produces a screening effect when the force on a star away from the origin is estimated. This can be calculated from the volume-distribution of dipole moment. The induced field is equivalent to an additional dipole

at the origin of strength

$$\sigma^2 p_i^{(1)} = - \frac{4\pi}{3} G_m N p_i. \quad (4)$$

Though time-dependent, it approaches a value of about 1/3 of the original dipole strength, in the opposite direction. But exactly as in the electrostatic case, this dipole field in turn reacts upon itself, and so on, until finally a value results that is the sum of a geometric series, as the final effective dipole moment. This series leads to an infinite-order differential equation; the reason for introducing σ as the operator of time-differentiation is that operational techniques provide a convenient way of handling it. Thus the dipole at the origin may be regarded as $1 + 4\pi G_m N / (3\sigma^2)$ times as large as the field it induces. (The operational method is useful to permit summation of this series of differential operators.) At short distances, a confusion analogous to that concerning "true" and "free" charges in electrostatics arises in the transition from an atomistic (independent-particle) picture to a continuum picture. Screening reduces the field by the factor

$$p_i \rightarrow p_i \left[1 + \frac{4\pi}{3} \frac{G_m N}{\sigma^2} \right]^{-1}. \quad (5)$$

The polarization produces forces at the center that tend to augment the original displacement, but the reaction of the system to the original particle is diminished by the screening. The reaction force may be calculated

by summing the force due to the (randomly located) dipoles. The stochastic sum is approximated by an integral. It is assumed that no force is generated by the displacement of the original star. The force per unit mass at the origin is

$$F_k = \int_L^\infty \frac{G P_i}{r^3} \left(\delta_{ik} - 3 \frac{x_i x_k}{r^2} \right) d^3 x / \left[1 + \frac{4\pi}{3} \frac{G_m N}{\sigma^2} \right]$$

$$= 8\pi G^2 m^2 N \frac{\delta_k^{(0)}}{\sigma^2} \frac{3}{4^3} / \left[1 + \frac{4\pi}{3} \frac{G_m N}{\sigma^2} \right]. \quad (6)$$

The lower limit L is a cutoff that is introduced to avoid the divergence at the origin; it may be taken to be about the mean near-neighbor distance, so $3/L^3 \approx 4\pi N$. The force is in the direction of the original displacement and is of magnitude

$$|F| = 2 (4\pi G_m N)^2 \frac{|\delta^{(0)}|}{\sigma^2} / \left[1 + \frac{4\pi}{3} \frac{G_m N}{\sigma^2} \right]. \quad (7)$$

A differential equation may be obtained by regarding the motion of $\delta_k^{(0)}$ as resulting from the force F_k :

$$\left[(4\pi G_m N)^2 \frac{d^4}{dt^4} + \frac{1}{3} (4\pi G_m N) \frac{d^2}{dt^2} - 2 \right] \delta_k^{(0)} = 0. \quad (8)$$

This has solutions of the form $\exp(\sigma t)$ with a positive real σ :

$$\frac{\sigma^2}{4\pi G_m N} = \frac{\sqrt{73} - 1}{6} = 1.257.. \quad (9)$$

(here σ is not regarded as the operator of differentiation). Because of the crude arguments involved, the precise value is probably not meaningful, but $\sigma^2 > 4\pi G_m N$. There is an exponential increase of $\delta_{\kappa}^{(0)}$ characterized by the free-fall time.

Even with screening, there is an exponentially growing root to the system of equations. The static approximation is analogous to the high-frequency limit in electrodynamics, where electron binding may be ignored. The appearance of a high-frequency limit is related to the fact that motions of individual stars are not taken into account, but were left out at the point where the variation of $\vec{r}$ in $D_{ij}(\vec{r})$ was ignored.

b) Low-Temperature Approximation

Stellar motions may be included in a low-temperature approximation by fixing stars to some equilibrium and position by elastic forces like those provided by the Lindblad elliptical epicycles and κ -oscillations (see Chandrasekhar, 1960; note that this picture implies a finite system). Although the anisotropy of the resulting "elastic constants" and the two dimensional (rather than three dimensional) character of the Lindblad orbits could be included without difficulty, the space will be treated as isotropic for simplicity; it can then be characterized by the epicyclic angular frequency κ . The dipolar force perturbs a set of (independent) harmonic oscillators, affecting the degree of polarization $\delta^{(1)}$, and modifying Eq. (9) for σ^2 :

$$\frac{\sigma^2 + \kappa^2}{4\pi G_m N} = \frac{\sqrt{73} - 1}{6} . \quad (10)$$

This equation has a real root for σ^2 as long as $\kappa^2/4\pi G_m N < 1.26$. In our galaxy, the vertical oscillation frequency $\kappa^2/4\pi G_m N < 1$, leaving a positive root for $\sigma^2/4\pi G_m N \sim 0.25$. The Lindblad picture thus slows the exponential departure, but only by about a factor of two. This is different in detail but not in principle from the findings of Toomre (1964) for a disk of stars. Suppression results if the elastic restoring force (reaction of the unperturbed medium on the perturbed orbit) exceeds the force due to the polarization.

c) High-Temperature Approximation

The situation in the solar neighborhood and in most stellar dynamical systems is one of high temperature. The Lindblad orbits are many times larger than mean near-neighbor distances.

In a high-temperature approximation, a star whips by the perturbed star rapidly and receives an impulse that depends on its precise path past the equivalent dipole and varies inversely with the velocity. The effect of the impulse is a displacement that grows with the time in such a way that the velocity factors cancel at a given distance. The polarization at some location is diminished by a dilution factor, but the region affected by the initial dipole is of correspondingly larger extension. (The effective solid angle for affecting stars coming into a given region is independent of the distance of the region from the perturbing dipole if orbits do not

re-curve and come back at the dipole from the other direction.) The picture is different in detail but the same effect persists.

d) Plane-Wave Perturbation

There is no need to restrict the perturbations to single stars; perturbations might be introduced to assemblies of stars or in patterns of convenient analytic form.¹ The dipole corresponds to a certain spherical symmetry. Another analytically simple pattern is the plane wave. Plane wave perturbations correspond to giving all stars of the second system a displacement relative to all stars of the first system by a small amount that may be characterized by a sinusoidal variation of amplitude $\vec{a}$ and wave number $\vec{k}$, generating a volume distribution of polarization varying with the same amplitude and wave number. The force per unit mass at some point can be calculated by evaluating an integral like that of Eq. (6) (without the screening term) and leads to forces in phase with the plane wave perturbation. The force always lies in the plane defined by the displacement and the wave vectors. The tensor relating the force to the displacement is diagonal in directions parallel and perpendicular to the wave-vector. The longitudinal (parallel to $\vec{k}$) force is related to the longitudinal displacement by

$$f = \frac{8\pi}{3} G m N a \quad (11)$$

¹ The system appears to be highly degenerate: almost any assumed set of displacements seems to lead to forces of the same general character. Normal modes, as such, have not been worked out.

in an obvious notation. The transverse component in the plane of $\vec{a}$ and $\vec{k}$ is $-1/2$ times as large. There is no force out of the plane.

By the same kind of static arguments as in subsection (a), this leads to exponential growth of the longitudinal mode characterized by the rate $\sigma^2 \sim \frac{2}{3} (4\pi G_m N)$, while the transverse modes do not show the growth. The force is independent of the wave number, and the phase factor carries through the calculation directly even if it is time-dependent.

The displacements of stars moving through this force-field on straight-line trajectories are easily calculated, assuming the force-field to be static. If these displacements are averaged over a (Maxwellian) velocity distribution, an average displacement results that is proportional to the imposed displacements:

$$\delta = \frac{4}{3} \frac{4\pi G_m N}{k^2 \langle u^2 \rangle} a \quad (12)$$

where $\langle u^2 \rangle$ is the mean square velocity in the direction of the wave-vector. If the induced displacement is larger than the original displacement, the displacements may grow, and a relation like the Jeans criterion emerges:

$$k^2 < \frac{4}{3} \frac{4\pi G_m N}{\langle u^2 \rangle} \quad (13)$$

The growth tends to be suppressed by the random motions, an effect also noted by Toomre (1964). A traveling plane wave causes no

complications; the velocities are simply replaced by the difference of particle and wave velocity; the stability criterion contains a factor $\exp(-v^2/\langle u^2 \rangle)$, where v is the component of phase velocity in the direction of the wave-vector. The growth rate is maximized with stationary plane waves.

III. DISCUSSION

The appearance of $(G\rho)^{-1/2}$ as a characteristic time is not surprising. With the quantities available, it is the only quantity of the proper dimensions that can be constructed, apart from dimensionless multiples. A dimensionless quantity can be constructed, but requires the introduction of something new:

$$Q = \left(\frac{LV^2}{G_m} \right), \quad (14)$$

where V is some characteristic velocity (such as the rms velocity). The relaxation time usually quoted is $(Q^{3/2} / \ln Q) \cdot (G\rho)^{-1/2}$ (Chandrasekhar, 1960). In the solar neighborhood, Q is a large number, on the order of 10^6 .

An essential feature of this approach is that a small displacement of one system relative to the other produces no effect. The internal structure must be disturbed; the disturbance is self perpetuating. While the picture presented is entirely in configuration space, the same behavior will occur in the associated velocity and phase spaces.

The answers to the questions posed in the introduction are now clear. The character of the disturbance surrounding a single perturbed particle is dipolar, falling off as r^{-3} . In a mathematical sense all stars are affected, but in an infinite system there is an effective range of a few times the mean near-neighbor separation and about 100 stars carry the principal involvement in the collective effects. In the static and low-temperature approximations the effects propagate instantaneously. If there were a step discontinuity in the dipole at the origin, the field would rise to its unscreened value and later drop off toward the screened value.

The effects seen in the earlier calculation here emerge in a system of many stars. The ergodic character of physical orbits should not be affected by phenomena of this kind.

The problem of including motions of other stars is complicated because contributions to the forces are most easily calculated for a volume-element that is fixed in a coordinate system co-moving with the original dipole. Stars move through this volume element. Extension of these arguments to a system of very few particles requires the calculation of expectation values that involve path integrals for each star in the system. An explicit comparison with the computer calculation (Miller, 1964) is therefore most easily handled as a numerical problem. Crude estimates indicate a very steep dependence on shape parameters. The estimates for infinite systems indicate that all stars in an n-body integration effectively participate until more than 100 stars are included.

Calculations like this cannot be expected to lead to solutions for equilibrium distributions, although they might lead to appreciation of the approach to equilibrium.

IV. EFFECTS OF THE POLARIZATION

While it is tempting to identify the growth of the polarization with a relaxation time, there are subtle differences of concept between a relaxation calculation and the present calculation. The relaxation time of interest to stellar dynamics is that according to which irregularities in the distribution function are removed--as with the re-establishment of the high-energy tail in the evaporation problem (Chandrasekhar, 1960). A system may have several relaxation processes, each with its own characteristic time (the nuclear spin system has at least three different relaxation times). Two explicit examples followed in some detail may illustrate the kind of effect predicted.

If a spherical velocity distribution with a cutoff (at the velocity of escape, say) is considered, stars may make transitions from inside to outside the sphere. A self-consistent solution with no evaporation has near it another solution leading to evaporation. The rate at which stars cross the sphere is given by the acceleration (i. e., force per unit mass) and the "density of states" at the surface of the sphere. The most naive attempt to calculate the rate at which stars cross the boundary (whether based on a finite discontinuity or on the velocity-space gradient of the distribution function) leads to logarithmically divergent integrals because

of the leading π^{-3} term in the force. However, it is clear that the arguments of this paper imply a rate of crossing of a spherical surface in velocity space (a rate of rearrangement of a velocity distribution) that is governed by the characteristic time according to which the polarization grows. The energies of individual stars change, but the total energy change within the system is zero.

Similarly, if the orbit of a star in a large system be considered, it has arbitrarily close to it another orbit from which it diverges exponentially. This applies as well to circular or epicycle orbits in the galaxy (the anisotropy of the local velocity distribution is small). In the absence of the polarization, epicyclic orbits would depart from each other with a linear or quadratic time dependence. The exponential is much faster. The stability of orbits in a stellar system is studied in terms of the smoothed potential (Chandrasekhar, 1960). The present argument implies that irregularities atop the smoothed potential override the smoothed potential and modify the orbits, although the restriction to small displacements does not permit any inferences concerning instability that are based on the notion of unbounded growth. In the usual separation of stellar dynamical potentials into smoothed potentials and a random fluctuation arising from binary encounters, the polarization will take over, producing a modification of the smoothed potential once a (random) binary encounter has generated a departure from the orbit of the smoothed potential. Because of the random initial condition, the motions resulting from the polarization have a quasi-random character, although in fact they reflect a systematic departure.

The polarization can magnify the effects of relaxation through its shorter characteristic time.

It appears that the characteristic time of the polarization enters the consideration of these two processes that are commonly regarded as being governed by "the relaxation time" in stellar dynamics. For each process, a system proceeding according to traditional ideas is found to have another system, arbitrarily close to it, that departs from it exponentially at a rate governed by the growth rate of the polarization. Still, the effect cannot properly be considered as a relaxation process, since these orbits can be time-reversible. It is not clear whether there is more than a superficial similarity between the two concepts.

Binary or ordinary multiple stars in the galaxy will not be seriously affected. The induced fluctuations in the force field have a scale on the order of a few times the mean near-neighbor separation. Since multiple systems are typically much closer, all members will feel about the same force. Systems whose components are so far apart that they are a sensible fraction of a mean near-neighbor distance will be affected.

Many of the arguments of stellar dynamics might be re-examined from this point of view. However, these effects only underline the statistical character of equilibrium in stellar dynamics.

This work was partially supported by the U. S. Atomic Energy Commission under Contract AT(11-1)-614.

REFERENCES

- Chandrasekhar, S. 1960, Principles of Stellar Dynamics (Dover, New York).
- Jeffreys, H., and Jeffreys, B., 1962, Methods of Mathematical Physics
(Cambridge University Press, Cambridge).
- Miller, R.H. 1964, Ap. J. 140, 250.
- Toomre, A. 1964, Ap. J. 139, 1217.

RELAXATION IN A MODEL CLUSTER

R. H. Miller

ABSTRACT

The interplay of the various factors affecting relaxation times in a star cluster is explored by integrating along orbits of different energies and angular momenta. Plummer's model provides the potentials for the orbits. Orbits of low angular momentum relax faster than orbits of the same energy with high angular momentum.

I. INTRODUCTION

The usual calculation of relaxation time in a stellar dynamical system is based on linear orbits in an infinite system (Chandrasekhar, 1960). If the system is not infinite and homogeneous some kind of averaging over actual orbits is implied. This can be handled after the manner of Michie (1963) by writing Fokker-Planck equations and seeking self consistent solutions, but his method does not yield a simple picture of the effect of relaxation on different orbits.

The present calculation was undertaken to illustrate the interplay among various factors that affect relaxation time. For example, an elongated orbit is expected to relax faster in the denser regions of a system notwithstanding the greater velocity dispersion there, but it spends much more time in the less dense regions, which leaves unclear the relative importance of relaxation processes in the different places. Relaxation is a "microscopic" process depending on local parameters, so total relaxation can be estimated by integrating (dt/T_r) along an actual orbit. The model cluster chosen for this process is Plummer's

(1911, 1915) model (polytrope of index 5), which has been discussed in detail by Eddington (1916). While perhaps not the best model that could be chosen for the purpose (see, for example, the work of Michie 1963 or of Limber 1961 for other possibilities), it is self-consistent under certain assumptions and greatly simplifies the calculation by permitting analytic treatment of some required properties. A similar calculation was reported by Woolley and Robertson (1956) for circular orbits and for zero angular momentum orbits in the isothermal sphere.

II. THE CALCULATION

Orbits in Plummer's model were integrated numerically. The potential(per unit mass) was normalized to $\phi = -(1+r^2)^{-1/2}$, so $\phi = -1$ at the center and goes to 0 as r^{-1} at large distances. Because of the spherical symmetry, orbits depend on two parameters: the energy $\mathcal{E}$, ($-1 \leq \mathcal{E} \leq 0$) and a parameter, α , that gives the angular momentum of an orbit as a fraction of the angular momentum of a circular orbit of the same energy. Since polytropic models are describable by distribution functions $f(\mathcal{E})$, all α ($0 \leq \alpha \leq 1$) must be equally probable. Taking P as the orbital period, the relaxation per unit time is

$$R = \frac{1}{P} \oint \frac{dt}{T_R} = \frac{2}{P} \int_{\text{I.T.P.}}^{\text{O.T.P.}} \frac{dr}{r T_R}, \quad (1)$$

in an obvious notation, with I. T. P. and O. T. P. the inner and outer turning points respectively. A trapezoidal rule was joined to analytic approximations at the (singularity at the) turning points for the integrations.

The period is determined by a similar integral. Orbits in Plummer's model range from simple harmonic near the center ($\mathcal{E} \approx -1$) to Keplerian far out ($-\mathcal{E} \ll 1$). Values for the radii of the turning points, radius and angular momentum of circular orbits, elapsed time, turning angle, and radial action integral were also generated.

Relaxation times were taken as functions of position (i. e., of R or of ϕ) within the model, according to the recipe of Chandrasekhar (1960; Eq. 2.379), reduced to a dimensionless form in which all numerical factors were suppressed. Thus the logarithmic term, which (indirectly) involves the particle number, was treated as substantially constant and ignored. The principal dependence in the relaxation time is through $\langle \sigma^2 \rangle^{3/2}$ and $N (= \rho)$. Values for $\langle \sigma^2 \rangle$ are proportional to $(-\phi)$. Values for N are proportional to $(-\phi)^5$ by the polytropic rule, so the relaxation time can be approximated (for very large total numbers of particles) by $(-\phi)^{7/2}$.

III. RESULTS

As might be expected, the fractional relaxation shows (integrable) singularities at both turning points when it is explored along the orbit. A more meaningful presentation is the relaxation per unit time averaged over an orbit, $\mathcal{R}$ of Eq. (1) (Figure 2; Figure 1 is given to show the orbit parameters). Although in arbitrary units, this gives the relative tendency of different orbits to relax. Actually, to the extent that the model is self-consistent, a detailed balance assures that relaxation does not affect the system. However, if some non-equilibrium model looks like this one (see, for example, Belzer, Gamow, and Keller 1951,

Michie and Bodenheimer 1963, Campbell 1966) the relative relaxation rates for low and high angular momentum orbits can be crudely compared--at higher energy, the lower angular momentum orbits relax noticeably faster than high angular momentum orbits.

This work was partially supported by the U.S. Atomic Energy Commission under Contract AT(11-1)-614.

REFERENCES

- Belzer, J., Gamow, G., Keller, G. 1951, Ap. J. 113, 166.
- Campbell, P.M. 1962, Proc. Nat. Acad. Sci. 48, 1993.
- _____ 1966, Proc. Nat. Acad. Sci. 55, 1.
- Chandrasekhar, S. 1960, Principles of Stellar Dynamics (Dover, New York).
- Eddington, A.S. 1916, M.N. 76, 572.
- Limber, D.N. 1961, Ap. J. 134, 537.
- Michie, R.W. 1963, M.N. 125, 127.
- Michie, R.W. and Bodenheimer, P.H. 1963, M.N. 126, 267.
- Plummer, H.C. 1911, M.N. 71, 460.
- _____ 1915, M.N. 76, 107.
- Woolley, RvdR., Robertson, D. 1956, M.N. 116, 288.

Fig. 1

Fig. 2

SOME COMMENTS ON GILBERT'S ANALYTIC GRAMMAR

Charles L. Robinson

1. Introduction

Gilbert [1] has presented an analytic grammar which under certain simple constraints is guaranteed to be unambiguous, yet, under more liberal constraints is the equivalent of a type 1 Chomsky phrase structure grammar. In this paper a different definition of an analytic grammar called an expanded analytic grammar is introduced with a new constraint parameter such that, while the additional constraint detracts nothing from the power of the model in general, it allows the grammar to correctly parse an arithmetic expression using the standard production rules [2]. Indeed, the context free portions of languages such as Algol can be represented in their traditional form.

Section 2 introduces the notation used in this paper. Section 3 then gives a critique of Gilbert's model and shows how the additional constraint solves some of the problems presented by that model. Finally section 4 discusses possible applications of this technique of language specification and implied compilation method.

2. Notation and Terminology

The reader is assumed to be familiar with Gilbert's basic idea of an analytic grammar specifying a "compilation" method for determining the "sentencehood" of an input string [1].

Let V be a vocabulary the members of which are called symbols

and are represented by lower case English letters. Two subsets of V are distinguished: V_T , the subset of terminal symbols, and $V_N = V - V_T$, the subset of nonterminal symbols. A string, represented by Greek characters, is a finite sequence of symbols $\Psi = v_1 v_2 \dots v_n$ and the length of a string $L(\Psi)$ is the number of symbols in the string. The null or empty string is given the name Λ ; and $L(\Lambda) = 0$.

A production is an ordered pair (Ψ, ϕ) of strings such that $L(\Psi) \geq L(\phi)$. A production is context free iff $L(\phi) = 1$. Expanding Floyd's notion [3] of an operator grammar we have: a production (Ψ, ϕ) is an operator production iff $\Psi = v_1 v_2 \dots v_j \dots v_n$, $v_j \in V_N$, implies that $v_{j-1} \in V_T$ and $v_{j+1} \in V_T$. The notion of the terminal vocabulary is purposely being left vague. Here we mean the set of all symbols from V which can appear in the original input string whose "sentencehood" is to be determined.

A production system (V, P) is a finite set P of production over a vocabulary V , and it is called unique iff for $(\Psi_1, \phi_1), (\Psi_2, \phi_2) \in P$, $\Psi_1 = \Psi_2$ implies $\phi_1 = \phi_2$. Just as the terms context-free and operator production applied to productions, the terms extend to production systems iff all productions within the system have the named trait.

Let $\mathcal{M}$ be the family of all finite sets M of integer quadruples (m, n, o, s) $1 \leq m \leq n$, $o = 0$ or 1 and $s \geq 0$. An expanded selection function F' is a computable function from $\mathcal{M}$ to $\mathcal{M}$ such that $F'(M) \subseteq M$

for every $M \in \mathcal{M}$. F' is called unique if, for every M , $F'(M)$ is either empty or consists of exactly one element.

Given a production system (V, P) an expanded production system $(\bar{V}, P', P'')$ is defined as follows. For every v_i in V both v_i and a new symbol $\bar{v}_i$, called a stopped v_i , are in $\bar{V}$. For all productions $P_i = (\Psi, \phi) \in P$ where ϕ is a string over V and $\Psi = v_1 v_2 \dots v_n$, let the following 2^n productions be in P'

$$\begin{aligned} & (v_1 v_2 \dots v_n, \phi) \\ & (v_1 v_2 \dots \bar{v}_n, \phi) \\ & \vdots \\ & (v_1 \bar{v}_2 \dots v_n, \phi) \\ & (v_1 \bar{v}_2 \dots \bar{v}_n, \phi) \\ & \vdots \\ & (\bar{v}_1 \bar{v}_2 \dots \bar{v}_n, \phi) \end{aligned}$$

For purposes of notation let z_i be either v_i or $\bar{v}_i$. Then we can simplify the last statement to read if $(v_1 v_2 \dots v_n, \phi) \in P$ then

$(z_1 z_2 \dots z_n, \phi) \in P'$. P'' is a unique set of productions of the form

$$(z_1 z_2 \dots v_r, z_1 z_2 \dots \bar{v}_r)$$

where there is a production

$$(v_1 v_2 \dots v_r \dots v_n, \phi) \in P.$$

The expanded scan function $S_{F'}$ is closely related to Gilbert's

scan function S_F . Let X be the family of all finite strings and let $(\bar{V}, P', P'')$ be an expanded production system. Let (ψ', ϕ') be a member of P' and (ψ'', ϕ'') be a member of P'' . Further let $x_1 \dots x_n$ be a string in X , $a_1 \dots a_r = \psi'$, $b_1 \dots b_z = \psi''$. The scan produces a set of quadruples in the following fashion. The quadruple $(e, e+r, o, s)$ is produced iff $x_1 \dots x_{e-1} x_e \dots x_{e+r} x_{e+r+1} \dots x_n = x_1 \dots x_{e-1} a_1 \dots a_r x_{e+r+1} \dots x_n$ and $s = j$, where j is the largest integer such that $x_j = \bar{v}_k$ for some $v_k \in V$ and 0 if no such x_j exists. The scan similarly produces a quadruple $(p, p+q, l, s)$ iff

$$x_1 \dots x_{p-1} x_p \dots x_{p+q} \dots x_n = x_1 \dots x_{p-1} b_1 \dots b_q \dots x_n$$

and s has the same value as above.

Given a production system (V, P) an extended analytic grammar is an ordered quintuplet $(\bar{V}, P', P'', z, S_{F'})$ where z , the selected symbol, satisfies $z \in V_N$ and if (ψ, ϕ) is in P then $\psi \neq \alpha z \beta$ for any strings α or β . By the definition of "sentencehood" below it is natural to restrict the production system such that if z is in ϕ then $L(\phi) = 1$, i.e., $z = \phi$.

A reduction of a string Σ is defined as follows. Let the scan function $S_{F'}$ acting on Σ pick a certain finite set $F'(M)$ of possible productions. One of the productions yielding a quadruple in $F'(M)$ is picked and this is applied to Σ forming Σ' . Diagrammatically let $\Sigma = \alpha \psi \beta$ and let the production (ψ, ϕ) yield a quadruple in

M which is subsequently found to be in $F'(M)$. (ψ, ϕ) is then a candidate for application to Σ and this application yields a new string $\Sigma' = \alpha \phi \beta$. For purposes of notation we write $\Sigma \Rightarrow \Sigma'$.

Σ is a sentence in the language represented by the extended analytic grammar $(\bar{V}, P', P', z, S_{F'})$ iff there is a finite series of reductions of $\Sigma = \Sigma^0 \Rightarrow \Sigma^1 \Rightarrow \dots \Rightarrow \Sigma^n$ such that $\Sigma^n = z$; or equivalently, since z can never appear on the left hand side of a production rule, if $F'(M)$ is empty after the i^{th} reduction then Σ is a sentence iff $\Sigma^i = z$.

If F' is equal to the identity function and the definitions of the productions allowed in P is suitably restricted and if P'' is empty then we have the class of Chomsky grammars. Note that if the productions allowed in P are again suitably restricted and either P'' is empty or F' is never a quadruple of the form $(m, n, 1, 0)$ then we have exactly Gilbert's analytic grammars.

3. The advantages of an expanded analytic grammar.

Henceforth we will deal only with unique P and unique $S_{F'}$. It is to be hoped that any description of a language be as "natural" as possible. This "naturalness" is primarily an aesthetic consideration tied closely with the attribute of "understandability," i.e., given a grammar G it should be easily deducible what strings of terminal characters are legal sentences of the language $\mathcal{L}(G)$ and vice versa. Unfortunately the analytic grammar of Gilbert does not possess this property even for context-free operator production systems.

For example let us consider the standard operator production system for arithmetic expressions. (For notational purposes let the production (ψ, ϕ) be represented by $\psi \rightarrow \phi$.)

$$\left. \begin{array}{l} i \rightarrow p \\ (p) \rightarrow p \\ p \rightarrow f \\ f \uparrow p \rightarrow f \\ f \rightarrow t \\ t * f \rightarrow t \\ t / f \rightarrow t \\ t \rightarrow e \\ +t \rightarrow e \\ e - t \rightarrow e \\ e + t \rightarrow e \\ e \rightarrow z \end{array} \right\} = P$$

$$\left. \begin{array}{l} V_T = \{ i, (,), \uparrow, *, /, +, - \} \\ V_N = \{ p, f, t, e, z \} \end{array} \right\} \checkmark$$

Let the analysis grammar be $G = (V, P, z, S_F)$ for any unique scan S_F and let the input string $i + i$ be scanned. It might be hoped that $i + i$ would eventually be reduced to Z and $i + i$ would be a sentence in $L(G)$. Unfortunately such will not be the case. The set of Gilbert pairs generated on the first scan is $\{(1, 1), (3, 3)\}$. No matter which pair S_F picks to apply, S_F must again pick the same pair if it is ever faced with the set $\{(1, 1), (3, 3)\}$ again. The reader can easily verify that the next set of pairs chosen is again the same as the first set and so on until we reach either $z + i$ or $i + z$.

That a unique scan analytic grammar can be constructed which will analyze an arithmetic expression is not denied, but it is clear that

the task is not an easy one. To show that this circumstance can be avoided in an expanded analytic grammar define $\bar{V}$ and P' as shown

in section 2 and include the following productions in P'' : If

$(v_1, v_2 \dots v_r \dots v_n, \emptyset)$ is in P then $z_1, z_2 \dots v_r \rightarrow z_1, z_2 \dots \bar{v}_r$ is in P'' iff some $v_1, \dots, v_r$ is non terminal and v_{r+1} is non-terminal.

In the case of arithmetic expressions this defines P'' to be

$$\begin{array}{lcl} f\uparrow & \rightarrow & \overline{f\uparrow} \\ t* & \rightarrow & \overline{t*} \\ t/ & \rightarrow & \overline{t/} \\ e- & \rightarrow & \overline{e-} \\ e+ & \rightarrow & \overline{e+} \end{array}$$

Let F' be the following "longest-leftmost" scan.

$F' = (m^*, n^*, l, s)$ if it exists, where m^* is the minimum m of all scan quadruples (m, n, l, s) and n^* is the maximum n of the scan quadruples (m^*, n, l, s) . If such a quadruple (m^*, n^*, l, s) does not exist, $F' = (m^*, n^*, 0, s)$ where m^* is the minimum m of all pairs $(m, n, 0, s)$ $n \geq s$ and n^* is the maximum of all quadruples $(m^*, n, 0, s)$.

(We can guarantee $F'(M)$ is nonempty for any nonempty M by running through this process again if the above quadruple did not exist, this time examining $(m, n, 0, s)$ and dropping the requirement of $n \geq s$. It does not make any difference in the case we are considering.)

We are now in a position to examine the proposed sentence $i + i$ in the grammar $(\bar{V}, P', P'', z, S_{F'})$. The series of transforms is as follows:

$$\begin{aligned} i + i &\rightarrow p + i \rightarrow f + i \rightarrow t + i \rightarrow e + i \rightarrow \\ e + \bar{i} &\rightarrow e + p \rightarrow e + f \rightarrow e + t \rightarrow e \rightarrow z. \end{aligned}$$

To take a more complicated example parse the sentence

$$i \uparrow (i \uparrow i) + i/i \uparrow i \uparrow i;$$

the steps in this parse are

$$1. p \uparrow (i \uparrow i) + i/i \uparrow i \uparrow i$$

$$2. f \uparrow ((i \uparrow i) + i/i \uparrow i \uparrow i)$$

$$3. f \bar{\uparrow} (i \uparrow i) + i/i \uparrow i \uparrow i$$

$$4. f \bar{\uparrow} (p \uparrow i) + i/i \uparrow i \uparrow i$$

$$5. f \bar{\uparrow} (f \uparrow i) + i/i \uparrow i \uparrow i$$

$$6. f \bar{\uparrow} (f \bar{\uparrow} i) + i/i \uparrow i \uparrow i$$

$$7. f \bar{\uparrow} (f \bar{\uparrow} p) + i/i \uparrow i \uparrow i$$

$$8. f \bar{\uparrow} (f) + i/i \uparrow i \uparrow i$$

$$9. f \bar{\uparrow} (t) + i/i \uparrow i \uparrow i$$

$$10. f \bar{\uparrow} (e) + i/i \uparrow i \uparrow i$$

$$11. f \bar{\uparrow} (p) + i/i \uparrow i \uparrow i$$

12. f + i/i ↑ i ↑ i
13. t + i/i ↑ i ↑ i
14. e + i/i ↑ i ↑ i
15. e + i/i ↑ i ↑ i
16. e + p/i ↑ i ↑ i
17. e + f/i ↑ i ↑ i
18. e + t/i ↑ i ↑ i
19. e + t/i ↑ i ↑ i
20. e + t/p ↑ i ↑ i
21. e + t/f ↑ i ↑ i
22. e + t/f ↑ i ↑ i
23. e + t/f ↑ p ↑ i
24. e + t/f ↑ i
25. e + t/f ↑ i
26. e + t/f ↑ p
27. e + t/f
28. e + t
29. e
30. z

Note: this reduction 18 $\Rightarrow$ 19 demonstrates that merely expanding P in the obvious way with productions from P'' is not sufficient for proper scan.

It should be noted that left recursion causes this "compiler" no trouble, and that the parse reflects the common semantic interpretation of the sentence.

It can easily be seen that no rules can be made for determining a

specific P'' and $S_{F'}$ for any general Chomsky grammar such that the expanded analytic grammar so defined produces an equivalent language. The fact that it is rather easy, with contrived examples, to "defeat" the choice made for scanning arithmetic expressions does not really indicate that the choices themselves were bad. In fact the choices seem to reflect the way humans scan such languages as ALGOL or FORTRAN. As an illustration both humans and expanded analytic grammars find it upsetting to analyze the following two examples in FORTRAN.

DONAMEI = 1, 10 (a do statement)

and DONAMEI = 1.10 (an assignment statement).

One additional constraint parameter could be considered, to wit an integer associated with each production which gives the function F'' acting on sets of quintuplets a chance to let one production dominate over another much in the same way that the productions of P'' dominate over productions from P' in the example given. If this separates non-unique elements of P then the requirement that P be unique can be dropped.

4. Applications

Wirth and Weber [4] claim quite reasonably that semantic rules should be intimately connected with syntactic analysis. The addition of such rules (such as the production of Polish suffix strings) are quite natural to this mode of analysis. Much work, however, needs

to be done in formulating these semantic rules. The main advantage of the expanded analytic scan is that the person writing the grammar knows exactly when each production of P' will be applied, and more important he knows that once a production is applied it will never be "undone," i.e. there are no trial parses or ambiguity modes [5] to worry him. All this seems to indicate that the grammar has a natural capacity to form a compiler-compiler.

As a compiling method the grammar is fairly fast if the scan and the function are combined. Many "programming efficiencies are obvious such as the application of productions from P'' while looking for the longest leftmost production from P' . Further, given a starting symbol, a reasonable search technique through the productions increases the speed tremendously.

REFERENCES

1. P. Gilbert, On the syntax of algorithmic languages, J. ACM 13 (January, 1966), pp. 90-107.
2. P. Naur (ed.), Revised report on the algorithmic language Algol 60, Comm. ACM 6 (January, 1963), pp. 1-17.
3. R. Floyd, Syntactic analysis and operator precedence, J. ACM 10 (July, 1963), pp. 316-333.
4. N. Wirth and H. Weber, Euler: A generalization of Algol, and its formal definition: Part I, Comm. ACM 9 (January, 1966), pp. 13-25.
5. J. Reynolds, Cogent Programming Manual, Argonne National Laboratory (March, 1965), ANL-7022.

THE PRODUCTION AND STABILIZATION OF REAL-TIME TASK SCHEDULES

G.K. Manacher

Abstract

A model for multiprocessor control is considered in which jobs are broken into various pieces, called tasks. Tasks are executed by single processing units. The structure controlling the assignment of tasks to processors in this paper is the task list, which orders all tasks according to servicing priority. When a processor becomes free, it simply picks up the highest priority task that is executable at that moment.

The job and its component tasks are imagined to be interacting with an environment consisting of a set of rigid timing constraints. Such constraints are of two types, called start-times and deadlines. The interaction is specified by requiring that certain distinguished tasks conform directly to one or more of these constraints. Tasks conforming to a start-time cannot begin until the start-time has passed, and tasks conforming to a deadline cannot proceed beyond the deadline. In our model, all tasks have known maximum run-times, but in any particular job execution, task run-times are unknown.

We show that despite the simplicity of this control scheme, some peculiar phenomena result. Most of these were first noticed by Richards.² A simulation study (Appendix A) reveals that these phenomena may be very common in practice. This paper and a forthcoming companion paper by R. L. Graham develop a coherent theory of task-list control in which the nature of these peculiarities is brought under systematic study. A number of potentially useful results are derived.

I. Introduction

By a multiprocessor computer we mean one in which several (we shall assume, identical) processing units, each one consisting of e.g., accumulators, index registers, etc. all have access to a common central memory. Such computers are of great interest, owing to the fact that at the present level of technical development, they are the chief hope for significantly increasing computation speed.

The usual kind of demand environment of a multiprocessor computer is statistical in nature -- we demand that a typical response to user requests be made in a "reasonable" time, and we tolerate small variations in service. Such response is typical of a time-sharing system. We shall call this kind of environment a "soft" real-time environment.

We are concerned in this paper with quite another kind of service, in which the external world presents the computer with a set of rigid timing bounds -- certain times before which it will not have data ready for processing, and certain other times beyond which it will not be able to make use of processed data. When the external world is characterized in this way, we will call it a "hard" real-time environment. The purpose of this paper is to investigate the mathematical properties of the simplest known control structure applicable to such an environment.

The structure whose properties we shall investigate controls the allocation of tasks to a set of processors, even when the run times of the various tasks, while bounded, are unknown. Furthermore it guarantees conformity to a "hard" real-time environment. The structure itself is a modified form of a scheme of B.P. Ochsner's called Task Assignment.¹¹ His essential idea is as follows. Programs are divided into segments called tasks; each task is executed by a single processor. It is in

general neither known nor cared which processor performs which task.

A program consists of one or more tasks; these tasks are linked together by precedence; task T_A must be done before task T_B , etc. Statements specifying such sequential relationships are called precedence relations.

We demand further that there exist no loops of tasks; that is, all programming loops must occur within tasks and not over tasks. The precedence structure of a program of tasks is therefore that of a partially ordered set. Hereafter, such a set will be referred to simply as a task set.

Ochsner has shown that the acquisition of an available task by an available processor may be conveniently accomplished through a structure known as a task list. By way of introduction, suppose Figure 1 is a "map" of our program. Time is assumed to flow from left to right.

At time 0 (vertical line) tasks T_1 , T_2 , and T_3 , the "initial" tasks, are known to be immediately executable. Task T_6 requires the completion of tasks T_1 and T_2 ; task T_7 requires the completion of task T_6 , and so forth. The entries in the task list are typically arranged in a linear order, as shown in table 1.

TABLE 1. Initial Configuration of Task List.

A	B	C	D
1	0	6	Addr. of task T_1
1	0	6	Addr. of task T_2
1	0	4, 5	Addr. of task T_3
0	1	-	Addr. of task T_4
0	1	-	Addr. of task T_5
0	2	-	Addr. of task T_6
0	1	-	Addr. of task T_7

The A field, called the "absolute enable bit field" tells, at any time, which tasks are ready for execution. The B, or "enable decrement field," tells how many of the tasks just prior to the task in question will have to be done. The C field or "successor" field, is a list (or, more likely, a pointer to a list) of tasks which immediately follow the task in question. Finally, the D field or "pointer" field, is a pointer to the program in core corresponding to the task.

The procedure for guaranteeing control is quite simple. A processor "enters" the list and scans it from the top down, looking for the first entry with the A field bit "up." It then "lowers" that bit, and enters the program specified in the D field. Upon completion of a task, it looks in the C field of the task. The B field of every task mentioned in this C field is decremented by 1, so that track is indeed kept of the number of antecedent programs yet to be done. When the B field of any entry reaches 0, the flag in the A field is put up, signalling that the task so flagged is ready for execution. The A field flag is lowered when a processor enters the program specified in the D field. Programs are terminated when no flag in the A field of any task is up, and no processor is busy. The reader is invited to run through Figure 1 using Table 1, to convince himself that the algorithm is valid.

Given a non-real time environment, the order of entries in a task list is not too important. Its only effect is on efficiency, and is usually not too severe. Mr. P. Bourdus* has shown, by means of a simulation employing a simulator designed and programmed by J.B. Little* that for typical task sets a program with a task list deliberately constructed with a view to minimizing

* Private Communication

the programs' run-time will typically run from 5 to 15 per cent faster than one with a task list ordered at random.

One may wonder just how much variability in total program run-time is possible over arbitrary variations in list order. The answer is given by R.L. Graham⁶: For n processors, the run-time can vary by a factor of $2 - 1/n$. This is a special case of his theorem, stated below.

To give the reader a feeling for these numbers, the run-time using a random list in the simulation mentioned above was 1.17 times its value when a carefully constructed list was used; the theoretical bound on this ratio, according to Graham, is $11/6 = 1.83$, since 6 processors were used in the simulation.

Varying the list order is not the only way in which peculiar increases in total run-time may be effected. Graham considers four types of changes which are all capable of providing such increases:

- 1) Changing the order of tasks in the list;
- 2) Removing some of the precedence relations;
- 3) Changing the number of processors, either increasing or decreasing it;
- 4) Shortening the run time of some tasks.

Cases 1), 3) and 4) above were apparently first discussed, qualitatively, by Richards.² Therefore, they will be called Richards' Anomalies.

Graham's contribution is a theorem which states that given two runs R and R', where R' is related to R by the application of any combination of the four conditions above, the total corresponding run-times ω and ω' are related to one another by Graham's Bound:

$$\omega'/\omega \leq 1 + \frac{n-1}{n'}$$

where n, n' are the number of processors in runs R and R' respectively. Graham's theorem also states that this bound may be arbitrarily closely approached when any one of 1) through 4) is applied, and that it is best possible in the sense that it cannot be replaced with a smaller number. It reduces to $2 - 1/n$ when $n' = n$.

Let us now suppose that the constraints on the tasks, the number of processors, and the task-list are fixed. We shall use the unqualified term "Richards' Anomaly" to mean an increase in total run-time produced by shortening the run-time of some tasks, even though this is only one of the four ways listed above in which "anomalous" behavior may occur.

In order to discuss the conformity of individual tasks to timing constraints, we must now alter our definition of Richards' Anomaly and in so doing stiffen it considerably. The reason for this is that in a complex system it is unrealistic to worry only about overall run-times. We may quite typically have a situation in which many tasks may have to meet absolute timing constraints even though they are not terminal tasks in the task set.³ We say that we have stabilized a task set in the weak sense when we have insured that its overall run

time will not surpass some limit, and we say that is stabilized or stabilized in the strong sense when we can insure that each task will be completed at or before a timing limit associated with that task. To this end, we state and prove an algorithm for adding a (usually) modest number of precedence relations to the original ones, such that stability is achieved without imposing unnecessarily severe constraints on the system. . A Richards' Anomaly is then said to occur whenever stability in the strong sense is violated.

We give in Figure 2 a simple example of Richards' Anomaly. Figure 2 is the same as Figure 1, but now the tasks in Figure 1 have an associated maximum run-time, shown as a subscript. A deadline at $t = 50$ is shown as a double vertical line.

Let the Task List L be $(T_1, T_2, T_3, T_5, T_6, T_7)$ scanned left to right. Given two processors, the program corresponding to the task set shown will run in 50 time units. The demonstration is by means of a Gantt Chart⁴, which is simply a time map of the activity of each processor. We display the Gantt Chart for Figure 2, given two processors called A and B, in Figure 3. The numbers refer to the task being executed.

The Richards' anomaly becomes apparent when we let t_3 , the time for execution of task 3, increase from 10 to $10 + \epsilon$ (ϵ is positive and arbitrarily small). In this case, the Gantt chart is shown in Figure 4.

This Gantt Chart is constructed according to the rule we referred to when discussing task-list assignments. Every time a processor becomes free, we choose the executable task that is highest on the list.

The anomaly is apparent: For $t_3 = 10 + \epsilon$, the total execution time is

45; for $t_3 = 10$, we get a total execution time of 50, assuming all other task run-times held constant.

We now demonstrate the fact that the meeting of deadlines may depend on Task-List order. Suppose task 4 (Figures 2 and 3) were required to finish by time 30. As things stand, Figure 3 does not satisfy this condition. The condition may be satisfied, however, by changing the task list to (1, 3, 4, 2, 5, 6, 7), in which case the Gantt chart is as shown in Figure 5.

A Gantt Chart constructed for a run in which all tasks run for their maximum run-times will be called a Standard Gantt Chart or SGC. A Gantt Chart representing a particular run in which each task runs for some arbitrary time, less than or equal to the maximum run time associated with that task, will be called simply a Gantt Chart. Thus every Gantt Chart has an associated (and known) SGC, namely the Gantt Chart that results when the same task set and the same Task List are used, but in which every task's run-time is replaced by its known upper bound.

The motivation for the study of Richards' Anomaly is provided in Appendix A, where it is shown by means of a simulation study that far from being a "mantlepiece curiosity," the occurrence of a Richards' Anomaly is in fact a very common phenomenon in multiprocessor systems.

The construction of "good" SGC's is a vitally important but somewhat peripheral issue to our discussion; it is discussed in Appendix B.

II. The Projective Task List.

Before discussing our algorithm for coping with Richards' Anomaly, we shall need to introduce one more idea, that of the Projective Task List

(PL). This is a particular task list that is in one-to-one correspondence with any Standard Gantt Chart. The PL is simply a list of all tasks ordered according to the time each task is picked up on the SGC, with the earliest-picked-up task first, the next-earliest next, etc. A convention regarding the definition of "earliest" is now clearly required in the case of ties, since quite commonly it will happen that two tasks are picked up at the same time. We resolve this ambiguity by imposing a rank-order on the processors. When two or more tasks suddenly become executable, the available processors (if any) scan the list simultaneously, and the first executable task is picked up by the highest-ranking free processor. The rest of the list is then scanned by the remaining free processors, etc. If several tasks are picked up at the same time, the one picked up by the highest-ranking processor is said to be the earliest, the one picked up by the next-highest-ranking processor the next-earliest, etc. When we draw Gantt Charts, we shall adopt the convention that the topmost line represents the highest rank-order processor, etc. Projective lists are then quite literally projective mappings of SGC's.

It appears to be the case that runs conducted with PL's tend to be freer from Richards' Anomaly than runs conducted with other lists yielding the same SGC. For example, if, in Figure 2, $t_3 \rightarrow 10 - \epsilon$, and we use its PL, the Gantt Chart becomes Figure 6 with execution time $45 - \epsilon$. A PL, however, will not always remove an instability. In Figure 7 suppose the task list is $(T_1, T_2, T_3, T_4, T_5, T_6)$; the reader may verify that this also happens to be the PL. Assuming two processors, the execution time is $2 + 2\epsilon$. If now either t_1 or t_2 becomes ϵ , the execution time becomes $3 + \epsilon$, demonstrating the instability.

III. Introduction to the Stability Algorithm

At this point it is convenient to introduce some notation. Consider a task set representation such as that of Figure 1. If some task T_B requires the completion of some task T_A , and if the head of the line representing T_A is directly connected to the tail of the line representing T_B , we say that T_A enables or precedes T_B and we write $T_A \rightarrow T_B$. If there exist some tasks $a_1 \dots a_n$ such that $T_A \rightarrow a_1 \rightarrow a_2 \rightarrow \dots \rightarrow a_n \rightarrow T_B$, we write $T_A \Rightarrow T_B$, and we say that T_B is a descendant of T_A and that T_A is an ancestor of T_B . The execution time for a task T_x is written t_x .

The stability algorithm is a constructive technique for guaranteeing that no task will ever be picked up at a time later than the time it is picked up on the SGC. It achieves stability by 1) requiring the use of PL's and 2) by introducing new precedence conditions of the form $T_A \rightarrow T_B$ for certain pairs of tasks which are related by the original precedence relations neither by $T_A \rightarrow T_B$ nor by $T_A \Rightarrow T_B$. Our scheme is relatively straightforward and results in a modest number of additional relations.*

*It should be mentioned here that there is a brute force solution to the problem which consists of forcing "serialization" of the SGC by this simple expedient: Suppose some task T_A is executed by processor X on the SGC; and suppose that there exists some task T_B that is picked up by processor X the moment T_A terminates. Then we let $T_A \rightarrow T_B$; the set of all such conditions obviously will suffice to stabilize the task set. This solution is poorer than that of the text because it fails to identify the source of instability. However, it does possess the virtue of working equally well with any task-list; in effect, such a heavy-handed procedure makes the task-list superfluous. In fact it amounts to the demand that each processor have its own task list, and thereby all the benefits of a collective task list vanish.

We present here our stability algorithm. This algorithm provides stabilization in the strong sense for any task set with respect to a SGC. (For the moment, we do not consider start-time constraints; these are discussed in Section IV.)

- (1) We are given an enabling structure in the form of a partially ordered set of tasks, together with maximum run-times for each and deadlines for some.

Via any procedure, construct a Gantt chart satisfying all precedence relations and deadline constraints, in which each task runs for its given maximum time. This chart is called the "Standard Gantt Chart" or SGC.

- (2) To form the projective task list, scan the entire SGC from left to right. Every time a new task is picked up by any processor, queue that task onto the list. If a group of tasks is picked up at a single point in time, queue the members of the group with respect to the rank-order of the respective processors that pick up each task. This rank-order is predetermined and fixed. The completed list is the projective task-list. (Another way to say (2) is that the task-list is the projection of tasks from the SGC.)

We need a definition at this point. The nearest common ancestor(s) of two tasks T_U and T_V are defined as those task(s) T_A for which $T_A \Rightarrow T_U$, $T_A \Rightarrow T_V$, such that there exists no task T_Q for which $T_A \Rightarrow T_Q$, $T_Q \Rightarrow T_U$ and $T_Q \Rightarrow T_V$. (That there can be more than one common ancestor of two tasks is shown in Figure 8, in which tasks T_U and T_V have nearest common ancestors T_A and T_B .)

- (3) Construct a second list, called the "pair list," two columns wide. The

first column consists of all pairs of tasks on the SGC which have a non-zero timing overlap. The second column contains the nearest common ancestor task(s) of the pair in the first column. An imaginary task I is postulated to "enable" all initial tasks.

- (4) For every ancestor task T_A appearing in the second column of the pair list, excepting I, consider all tasks π_A that are members of pairs in column one whose nearest ancestors include T_A . The task in π_A with the earliest starting-time is called T_A^* . All other tasks are called "left" or "right" of T_A^* respectively according to whether they start before or after T_A^* [see Section II for what happens in case of ties].
- (5) Every "right" task T_R having only "left" parents (including possibly I) enters into a new precedence relation, known as a stability condition, of the form

$$T_A \rightarrow T_R \quad (I)$$

which must be added to the already existing set of precedence conditions, unless it is already the case that $T_A \Rightarrow T_R$.

We now state our Stability Theorem:

The set of all enabling conditions (I) will stabilize a task set in the sense that it is sufficient to guarantee all deadlines regardless of time-shrinkage of any number of tasks, provided that the SGC meets all deadline.

Let us try out our algorithm on Figure 7, using the task list $(T_1, T_2, T_3, T_4, T_5, T_6)$ as before. The SGC is shown immediately below, for 2 processors.

The pair-list is as follows:

TABLE 2

<u>Co-running Pair</u>	<u>Common Ancestor(s)</u>
1, 2	I
3, 4	1, 2
5, 6	I

In table 2, there are 2 common ancestor tasks (save I), namely T_1 and T_2 . Tasks T_3 and T_4 have T_1 and T_2 in common as nearest common ancestors. T_3 is the "earlier" of T_3 and T_4 , since it is picked up by a higher-ranking processor. Therefore $T_1^* = T_2^* = T_3$. The "right" tasks, relative to T_3 , are T_4 , T_5 , and T_6 ; the "left" tasks are T_1 and T_2 . T_6 has only a "left" parent (I), so stability conditions $T_1 \rightarrow T_6$, $T_2 \rightarrow T_6$ must be introduced. These, by (5) above, are sufficient to stabilize the lattice, as the reader may verify.

III'. Proof of the Stability Algorithm; Formal Specification of the Problem.

(This section is contained in Appendix C; it may be skipped if the reader wishes.)

IV. Multiple Start-Times and their effect on stability.

We have shown how to achieve stability when all tasks have a common start-time; however, for some applications this may be very unrealistic. In "real life", one might easily expect task sets which have the appearance of Figure 10.

The new element in Figure 10 is the appearance of start-time y , supposed fixed in time. If a given start-time is not the earliest one, we refer to it as subsidiary. Such a case might occur, for example, on a work project, when certain supplies are not scheduled to arrive until some time after the project has begun. In a real-time computer program, they might correspond to inputs supplied at various discrete times within some timing template by the external hardware, which sends a "go" message when it has data ready.

The handling of multiple start-times is necessarily a more unpleasant business than the corresponding handling of multiple deadlines. The reason for this is not hard to guess. The tasks enabled by the "primary" start-time, e.g., x in Figure 10, may occupy processors in a highly unpredictable way, making for significant delays in the picking-up of tasks enabled by subsidiary start-times. An easy way to see how "abnormal" the behavior of a task system becomes under these conditions is to observe that an instability may occur when multiple start-times are allowed, even when there is only one processor and a projective task list is employed. As an illustration, consider the task set of Figure 11 and the corresponding SGC of Figure 12. The PL is (T_1, T_3, T_2) .

If t_1 now becomes $1 - \epsilon$, the SGC will be as shown in Figure 13, demonstrating the instability.

Three approaches have been discovered for handling this problem; each has limitations; however, all have associated stability algorithms. The first method adds stability relations beyond those given in the stability algorithm. The second associates each subsidiary start-time with an interruption of the system. The third imposes a fixed upper bound on the maximum run-time of any task.

Each of the approaches begins with a Standard Gantt Chart. (The presence of multiple start-times does not alter the interpretation of the SGC.)

(A) In the first method, we regard each subsidiary start-time as a null common ancestor task, similar to T_A in the stability algorithm, even though it is not really a task. This null "task" is represented on the SGC as a point corresponding to the actual start-time. It is allowed to be represented as if it sat "on top of"

a real task that is running at the time the start-time occurs. Which processor it "sits" on is immaterial. We achieve stability by imagining that subsidiary tasks play the role of "common ancestor tasks" T_A in (4) of the stability algorithm. The earliest-starting task that is enabled by the subsidiary start-time plays the role of T_A^* , regardless of whether or not it has timing overlap with another task sharing the same property.

In Figure 14 is shown an SGC corresponding to the task set of Figure 9. The arrow on Figure 14 corresponds to the start-time y of Figure 9. In this example, stabilization is achieved by introducing the stability condition $y \rightarrow T_7$. Thus, task T_7 cannot start until 1) task T_2 is done and 2) start-time y has passed.*

(B) In the second method, we regard every subsidiary start-time as an interrupt, which demands that every processor stop what it is doing and rescan the task list, which at that point includes among its executable tasks those tasks that were just enabled by the subsidiary start-time. Tasks that are interrupted and not immediately resumed are "reactivated" in that their A flag is put back "up". When a processor later reenters a task that was cut off in midstage in this way, it begins where the last processor to work on the task left off.

* In a sense a heavy price is paid for this procedure. Up to now, we have added stability conditions which are of the form $T' \rightarrow T$. Such stability conditions would not hurt us if, for instance, all task run times were to shrink to half of their maxima: The task set would run in half the time. The delays imposed by requiring that a task await a subsidiary start-time are far more damaging, since they can effectively counteract the advantage that would otherwise accrue from massive run-time shrinkage by imposing absolute, rather than logical, timing constraints.

(C) A third approach - probably the most well-known - is via a notion that has been called "granularity." The idea is straightforward: We impose an upper bound M on the maximum run-times of all tasks; this is called the "grain size". We then know that if a task is enabled by a subsidiary start-time, at most M units of time can elapse before a processor will become free.

In this scheme, the SGC is constructed subject to the constraint that all tasks enabled by a subsidiary start-time σ must be placed on the SGC in such a way that their start-times are $\geq \sigma + M$.

The result of following the above rule is that no subsidiary start-time enters into a stability condition; this overcomes the objection raised in subsection (A).

The disadvantage of the various schemes are clear. In (A), we tend to slow thing down; we keep some tasks waiting until some start-time has passed. The result is that processors may be unnecessarily idle. In (B), we have the burden of handling broken tasks; and in (C) we restrict task length, which may cramp programming style.

V. Multilist Timing Control

Consider the problem of handling many jobs, each in the form of a task set. Suppose the jobs do not interact. This means that no task in a given task set is a descendant of a task in another task set (see Figure 5). Suppose these task sets each come with a "private" task list, and for some reason (lack of a computation facility, of time, etc.) we cannot integrate them by constructing a collective task list. The situation can still be controlled, although some inefficiency will result.

An appealing way to do this is to assign a fixed number of processors to each non-interacting task set. For instance, we might assign 2 processors to task set 2 in Figure 15, and 1 processor to task set 1.

We now adopt a simplified representation of task sets. Consider for a moment only the initial start-time and final deadline as "bounding" a task set. Then a task set might typically be represented as in Figure 16, where we have shrunk all tasks to a single line. The single vertical line represents the earliest start time; the double line, the latest deadline for the task set. The numbers are to be understood as follows: Given 4, 5, 6... processors, the time to execute the task set is the time from the single vertical line to the first arrow.* Given 3 processors, the time to execute the task set is the time from the single vertical line to the second arrow, and so forth. Two processors yield an overrunning solution, as does 1 processor. (The reason that 4, 5, 6... processors all give the same run-time is typically that this run-time is equal to the longest sequential path through the task set, and no improvement is possible.)

The number of processors it is useful to consider is thus clearly bounded, on the high side by critical-path considerations and on the low side by overrun considerations.

It will be sufficient, then, to represent, e.g. Figure 16 by Figure 17, since all other information is extraneous. If we now have many task sets, they may be represented together as an ensemble such as is depicted in Figure 17.

*In this section, timing refers to SGC's.

Figure 17 shows what shall be called a task set ensemble. If it is decided to run it by assigning a fixed number of processors to each lattice, we must remember that we cannot allocate more than the total number of processors at hand. In practice, this means that a particular number of processors must be picked for each task set within the limits shown, as in Figure 18, such that no vertical line ever "intersects more processors" than we actually have. In Figure 18, for instance, if we have only 6 processors, we can succeed by particularizing it as shown in Figure 19.

No algorithm is known to the author for carrying out transitions from the "generalized" task sets of Figure 18 to the "particular" of Figure 19 in such a way as to minimize the number of processors required. However, a sub-optimal technique for a very similar problem is discussed in reference 7.

We note in passing that enabling structures similar to task sets can be set up, whose basic elements are whole jobs rather than single tasks. Thus job J_2 cannot be done until job J_1 has been done, etc. However, the resemblance is superficial: The jobs are assigned a fixed group of processors, and are enabled by a system clock. The advantage to be gained by such a procedure is that jobs can be switched in the sense that a given job in a multijob schedule can be replaced at run time by another that is known* to run in the same or less time, given the same number of processors.

Multi-list control is thus strikingly similar to multiprogramming control¹⁰ where processor allocation is handled in the same fashion as memory allocation in a multiprogramming system.

*This knowledge is available precisely because each job corresponds to a stabilized task list.

VI. Some Points Concerning Control and Some Speculations.

The existence of good stable schedules with guaranteed deadlines, and algorithms for constructing them, means that we can expand the customary notion of a loader, which is a device for "scheduling" single-processor tasks spacewise into memory. We may conceive of a space-time loader whose job it is to pre-allocate both space and time for the various "assemblies," i.e. tasks, which are the object of its manipulations. It might even allocate processors for jobs consisting of task sets. Just as it is possible to communicate program space requirements to a conventional loader, so it should be possible to communicate timing requirements to a space-time loader.

Via pseudo-ops, a skeletal language could be provided in which timing information and precedence relations between tasks are easily expressed. This information would be used for automatically scheduling tasks.

The basic timing information that accompanies each task might consist of:

- a) A statement (obligatory) giving the maximum run time. During execution this would set the Δ -clock (a clock generating an interrupt after a certain interval of time has elapsed).
- b) A statement giving the name or value of the deadline and/or start time, where applicable to the task.
- c) A declaration indicating which location is to be transferred to on interrupt from the Δ -clock.
- d) A means for testing the Δ -clock.

e) Instructions for indicating, within the Δ -clockout trapping routine, what action is to be taken: should the task be terminated? Should further time be allocated? How much? Should its successors be enabled or aborted? Should an error message be printed?

The basic linkage information accompanying each task consists simply of a declaration which identifies the tasks that are "parents" and "sons" of the task.

Turning our attention to interrupts, we notice that a system such as ours will require that each processor be able to have considerable flexibility in determining which interrupts it is to be responsive to. For instance, if we assign n processors to one hard real-time job, we will want these and only these processors to respond to an interrupt sent by a peripheral device associated with that job.

Finally, we notice that even though a group of processors is committed to a particular "hard" real-time job, it may still be possible to use these processors simultaneously to provide admittedly poor-quality "soft" real-time service. We require that each task's representation on the task list have two fields in addition to the four we have discussed so far. The first additional field, the E field, contains the start-time of the task on the SGC. The second, the F field, has one bit and this bit, originally 0, is set to 1 when a processor enters the program specified by the task (the address in the D field). The bit then remains 1.

The procedure is as follows. Suppose a user on the outside wants service. He indicates this, and at the next occasion that a task terminates, the processor

freed scans the task list, looking for the first task whose F field is "down".

It obtains the value in the E field of this entry, and subtracts from it the current time, obtaining a time-quantity δ . If δ turns out positive, it appears to be plausible that the processor can "lend itself" to an outside user for a time $\leq \delta$ without unstabilizing the hard real-time process it is principally engaged in.

Attempts at a proof for this speculation have been unsuccessful. However, some guidelines along which we believe a proof can be constructed are given in Appendix D.

VII. Priority vs. Quick Execution and Stability

The phrase in general use expressing the fact that T_A is above T_B in a task-list is " T_A is of higher priority than T_B ". This way of talking assumes (correctly as a rule) that T_A will usually get done ahead of T_B . According to this view, task-lists are to be constructed so that "important" tasks are on top and "unimportant" one are on the bottom. This strategy does not correspond to our "projective list" construction. There is a situation, however, in which task-list ordering must be considered from both a "stability" and a "priority" point of view. This is the situation in which we have more tasks to do than can be guaranteed; that is, where there exists no way to map the tasks onto a Standard Gantt Chart so that they all get done on time. We call this an "overflow situation." Stability is as important in this "overflow situation" as it is in the situation discussed previously. The reason is quite obvious. Without stability, we know about the probability of execution of a task only in a rough sort of way. Let X be the position of a task on a list (increasing X means going

down the list). The probability of execution will be as in Figure 20. Suppose, however, that we construct a Standard Gantt Chart in such a way that we cause high priority tasks to be done within the deadline constraint; other tasks are mapped as if they are run after the deadline; we call them "outsiders". Imagine that the "outsiders" are priority-ordered; the later a task gets done on the SGC, the lower its priority.* The situation is as in Figure 21.

Now let us stabilize the task set, using Figure 21 as a basis, extending the stabilization procedure to cover all tasks, including the "outsiders". The probability curve is now as in Figure 22.

The flat part of the curve corresponds to our previous discussion. The "tail" shows the expected behavior. The guaranteed part is stabilized. The reason for extending the stabilization procedure to tasks lying beyond the deadline on the SGC is as follows. A non-guaranteed task whose end falls just beyond the deadline has a good chance of being executed if some of the guaranteed tasks run for less than their maximum run-times. Let us quantify this idea. Suppose at some time t_1 on the SGC, a subset ϕ of the tasks has been run. Suppose now that during a typical run it turns out to be the case that at $t_2 (< t_1)$ the subset ϕ has run. Clearly, we can imagine this as equivalent in a sense to a time-translation of $\alpha = t_1 - t_2$ units of time on the SGC. Suppose now that there is some non-guaranteed task whose SGC end-time differs from the deadline time by less than α . Since the stabilization procedure is invariant under time-translation, it follows that this non-guaranteed task now will definitely run

* We are still using a Projective List.

providing that the stabilization procedure extends over all tasks, not merely the guaranteed ones.

Finally, consider what we can do when the "raw" priority ordering is given to use in the form of a task list rather than an SGC. (This is a reasonable expectation.) The task-list will almost certainly not be projective as given. The technique is simple:

1. Using the given task-list, construct an SGC by letting each task run for its maximum time.
2. Obtain a new, projective task-list from this SGC, and stabilize it.
3. When actually running, use the new task-list.

VIII. General Comments and Remarks on Boundary Conditions

A remark may be made here concerning a connection between the Graham's Bound discussed in Section I and a simple rule of thumb. Suppose we are interested only in overall timing of a lattice, i.e. with single deadlines for whole task sets. Suppose we wish to have complete freedom to shorten tasks, change task lists, or even lose a processor without having to worry about Richards' anomaly. Is this possible?

Since for shortening of tasks and/or list changes, Graham's bound gives $\omega'/\omega \leq 2 - 1/n$, and for these changes together with loss of a processor gives $\omega'/\omega \leq 2$, we have an absolute bound on "anomalous" behavior. Suppose,

given a task set, there exists some task-list such that all the tasks, running for their maximum run-times, get done in half or less than half of the time allocated to them. In this case, we need not introduce any stability procedure; we may change the task-list arbitrarily by rearranging tasks, and we can even afford to lose a processor. We will never run over the deadline. It is only in the case that things are "tighter" than this that we must worry about stability, losing a processor, etc. Graham's Bound, incidentally, is not applicable to any deadlines other than the overall deadline; in particular, it does not apply to deadlines for single tasks.

An unanswered question of some interest has to do with granularity (Section 4C). It is known that Graham's bound is not changed by constraint that all tasks be forbidden to run longer than some fixed quantity. Thus, counter-intuitively, imposing granularity does not affect instability limits. It nevertheless seems intuitively reasonable that in some sense, the more microscopic the granularity, the less likely it is that Graham's bound can be attained in a typical run. This question clearly requires a statistical approach.

IX. Conclusions

We are able to construct, internally and without programmer intervention, near-optimal task-lists for almost any feasible combination of task run-times, deadlines, and/or external enabling conditions. We are able over a wide range of situations to guarantee that every deadline will be met. When our schedule is "full" in the sense that we are up to capacity with guaranteed tasks, we can still take on extra tasks whose execution is not guaranteed, but whose presence does not alter the assurance that the guaranteed tasks remain guaranteed.

It is suggested that a monitor technique is in sight, in which a "time-space loader" non-dynamically handles both time and space integration of single-task programs into a real-time, multiprocessor system.

X. Acknowledgments

E.N. Gilbert had helpful suggestions. R.L. Graham's insights proved enormously valuable. M.D. McIlroy pointed out errors in an earlier edition. P. Bourdus graciously made his services available for simulation runs, and Rudd Canaday and most particularly Will Renkel provided theoretical insights for which the author is thankful. B.P. Ochsner's constructive criticism has been of steady value.

All of these individuals are from Bell Telephone Laboratories, Inc.

APPENDIX A

Results of a Simulation

In order to test our stability algorithm and the SGC-forming procedure discussed in Appendix B, and to estimate how commonplace Richards' Anomaly² is in the "real world," we used a complicated 85-task set called "roadmap" which P. Bourdieu had been using as a guinea pig in his simulation studies. A picture of this task set is in Figure 23. In it, all task run-times are taken to be 1 time unit.*

By means of the technique described in Appendix B, SGC's were obtained for $n = 2, 3, \dots, 9$ processors. As it turned out, these SGC's were such as to absolutely minimize run time for the task set of Figure 23.

Richards' Anomaly was tested for by changing the task running times. In the first trial, the run time for all tasks was $.995 \pm .005$ time units. The probability was uniform over this interval and zero outside of it. Within this range, run times were assigned to tasks in random fashion, e.g. $t_1 = .997$, $t_2 = .998$, $t_3 = .995$, etc. The task-list was unchanged. Forty runs were made. The results are in Figure 24, which shows most effectively the destructive potential of Richards' Anomaly. Note that about 4/5 of the runs shows "anomalous" behavior.

In Figure 25 is shown the same situation for a timing distribution of $.95 \pm .05$, and in Figure 26, for $.67 \pm .33$. Figure 26 shows that a sufficient amount of randomly distributed shrinkage will eventually overwhelm the anomaly. Figure 27 is shown for comparison.

* Admittedly, making all run times equal has the effect of amplifying Richards' Anomaly.

Stability was achieved by employing our stability algorithm. Some 50 enables were added to the existing 155.

More enables (76, in fact) are actually predicted by the algorithm, but some of them were then removed as unnecessary by using the conjecture discussed at the end of Appendix C. The results indicate that this conjecture may be valid.

APPENDIX B

The construction of SGC's which are Near-Optimal with Regard to Timing*.I. The simple case of a single deadline.

There is an approach to this problem in the form of a heuristic which appears to have the ability to generate task lists which are almost always optimal, and which are very "good" even when not quite optimal.⁹ Some preliminary discussion will help.

Mr. W. Renkel[†] first suggested to me the importance of operation-theoretic methods in problems of this kind. In particular, a quantity of central importance is the critical path. In a task set such as we have been describing, we can think of the tasks as lines, and their intersections (including the terminal points) as nodes.** The critical path between two nodes is the longest path between the nodes, assuming that it is possible to travel from one node to the other.

For example, in Figure 28 the critical path from point A, at the end of task 1, to point B, at the end of tasks 5 and 6, is of length 50, since there exists a path (via tasks 4, 5) which takes that long to traverse. (Note that we always respect the directional sense of the tasks in examining possible paths through the task set). The alternative paths from A to B are via tasks 2, 3, 4 taking 45, and via task 6, taking 40. Since of all available paths, that leading through (4, 5) takes longest, that path is identified as the critical path. The critical time, corresponding to the critical path from A to B is denoted $T_{crit_{AB}}$.

*It should be noted that there may be many criteria which may partially dictate task-list order; this discussion deals only with the problem of timing.

**After this paper was written, it was pointed out to the author that the reverse strategy is actually better, since it guarantees that any loop-free enabling network can be represented as a partially ordered set.

[†]Present Address: Bell Telephone Laboratories, Whippany, N.J.

In order to simplify notation, we adopt the convention of specifying a node by the termination of a task. Thus node A can be referred to as node 1, node B as node 5 or node 6, etc.

The following propositions are easily deduced and therefore presented without proof:

1. Given that task Q has just been accomplished, an interval of at least $T_{crit_{QR}}$ will be required before task R is accomplished, regardless of the number of processors available to the job.
2. Given a deadline T_e , the latest permissible starting time for any task i is

$$\Delta_i = T_e - T_{crit_{ie}} - T_i$$

where $T_{crit_{ie}}$ is the critical time from the i-th task (i.e. from the end of the i-th task) to the deadline, and T_i is the maximum execution time for the i-th task.

The times referred to as characteristic of a particular task are the maximum estimated run times, since we are concerned in this appendix only with SGC's.

A reasonably near-optimal means for specifying the assignment of tasks to available processors may now be given. Let us suppose that at some time, k tasks are ready for execution. Clearly, that task among the k for which Δ_i is minimal is in some reasonable sense the most "urgent" task, and should be chosen. We call this construction rule the "urgency" strategy.

Let us see what this means for Figure 2. First of all we make up an "urgency table," i. vs Δ_i ; for purposes of introduction, we also show the

intermediate steps:

TABLE 3

Task	T_i	$T_{crit_{ie}}$	Δ_i
1	20	20	10
2	10	20	20
3	10	15	25
4	15	0	35
5	10	0	40
6	15	5	30
7	5	0	45

According to the urgency strategy, the standard Gantt Chart can be constructed as shown in Figure 29.

Notice that there is a slight ambiguity when two tasks end at the same time as to which processor is assigned to the most urgent available task. In such cases we arbitrarily let processor A be of highest urgency, B next, C next, and so forth, so that we can establish a diagrammatic convention of "topmost processor first" on the Gantt Chart. Thus in Figure 7 task 6 was chosen before task 4 owing to $\Delta_6 = 30$, $\Delta_4 = 35$; and it was assigned to processor A rather than processor B.

II. The handling of multiple deadlines.

Consider the task set shown in Figure 30.

The essentially new features are these: 1) The end tasks - those that enable no task - namely tasks T_5 , T_7 and T_8 , now meet different deadlines; task T_7 must be done by time 50 while tasks T_5 and T_8 must be done by 45. 2) An intermediate task (task T_4) must also meet a deadline; task T_4 enables task T_8 and must be done by 30.

The heuristic we use for multiple end-times is only a slight generalization

of the previous method. We associate an index j with end-times. Thus, in our example we might have end-time α associated with $j = 1$, β with $j = 2$, and so forth. The j th end time we call e_j .

We say that a task g is part of the j th task subset when there exists some path from (the end of) g to e_j , or, to put it more accurately, to some task on whose end e_j lies. (We abbreviate this $g \Rightarrow e_j$). Tasks that must meet explicit deadlines we call "deadline tasks"; in Figure 30, they are tasks T_4 , T_5 , T_8 , and T_7 . For example, in Figure 30, if $e_1 = \alpha$, $e_2 = \beta$, and $e_3 = \gamma$, then $T_1 \Rightarrow e_2$, $T_3 \Rightarrow e_1$, but $T_3 \not\Rightarrow e_2$.

With reference to Figure 30, task subset membership is as follows:

TABLE 4

<u>End-Time</u>	<u>Task Subset Member Tasks</u>
α	3, 4, 5, 8
β	1, 2, 6, 7
γ	3, 4

The scheduling technique is as follows. We define the critical path

$T_{crit_{ij}}$ from the i th task to the j th end-time as the longest path from i to e_j .

This definition of course holds provided that i is a member of the j th task subset,

else $T_{crit_{ij}}$ is undefined.

Now define

$$\Delta_{ij} = \begin{cases} T_{e_j} - T_{crit_{ij}} - T_i & \text{when } i \text{ is in } j\text{th task subset} \\ \text{undefined} & \text{otherwise} \end{cases}$$

The Δ_i is now defined by

$$\Delta_i = \min_j (\Delta_{ij})$$

and the heuristic for scheduling remains as before.

For Figure 30, our computation is summarized in Table 5:

TABLE 5

Task	$T_{crit, i\alpha}$	$T_{crit, i\beta}$	$T_{crit, i\delta}$	$\Delta_{i\alpha}$	$\Delta_{i\beta}$	$\Delta_{i\delta}$	Δ_i
1	-	20	-	-	10	-	10
2	-	20	-	-	20	-	20
3	20	-	15	15	-	5	5
4	5	-	0	25	-	15	15
5	0	-	-	35	-	-	35
6	-	5	-	-	30	-	30
7	-	0	-	-	45	-	45
8	0	-	-	40	-	-	40

For two processors, the Standard Gantt chart is as shown in Figure 31.

It may be verified that all deadlines have been met. Note that there is in the task-list a commingling of tasks from various parts of Figure 30 such as to make one suspect that task-list construction by non-algorithmic methods cannot compete with the more mathematical procedure outlined here.

It appears then, that there exists a method for the production of good SGC's given that all tasks have known maximum running times, and given deadlines for some tasks. No easy method is known⁵ for producing optimal lists, and it is not even easy to define optimality for the case in which there is more than one deadline, since one must then weigh the respective margins by which each deadline is "made."

For the case of a single deadline, a program has been written in SNØBØL⁸ for the production of SGC's by the procedure outlined in this appendix, given precedence relations and run-times for an arbitrary number of tasks. (Extension of this program to the case of many deadlines would pose no serious

problem). The program was tested out on Bourdus' "Roadmap", consisting of 85 more or less randomly connected tasks. Bourdus had already used a "hand" method to construct a task list; he compared running times given this list to running times for a "Random" list. "Roadmap" is shown in Figure 23. The results of the SNØBØL run are summarized in Table 6, which displays timing for SGC's constructed by various means and provides an example of the strength of the minimum - Δ_i strategy discussed in the text.

TABLE 6. Various Execution Times for Executing the Tasks in "Roadmap".

No. of Processors	Min - Δ_i Strategy Algorithm	Fairly Shrewd Guessing	Random
3	29	29	30
4	22	22	26
5	18	19	22
6	15	17	18
7	14	16	17
8	14	14	15
9	14	14	14

That the critical-path method is not optimal may be seen from Figure 32.

The critical-path method yields a run-time of 30; however, with a bit of ingenuity, the reader may produce an SGC for which a run-time of 29 is obtained.

APPENDIX C

(A) Proof of the Stability Theorem

We formalize here some of our notions: We are given n identical processing units P_i ($i = 1, 2, \dots, n$) and a set of tasks T_j ($j = 1, 2, \dots, m$) to be run on the P_i . There is also given a partial order¹ on the T_j and a time t_j which task T_j takes to run. A processor, having begun some task T_j , works on it without interruption until its completion. Since the tasks are partially ordered, they may be conceived of as points in a Partially-Ordered set, i.e., a directed graph with no loops. If there is a directed arc from T_j to T_k , we write $T_j \rightarrow T_k$. T_j is then said to be the parent of T_k and T_k the son of T_j . When all the parents of T_k have been completed, T_k is said to be executable.

If there is some sequence $T_j \rightarrow T_{j_1} \rightarrow T_{j_2} \dots \rightarrow T_{j_n} \rightarrow T_k$ [$T_{j_1} \dots T_{j_n}$ possible null] we write $T_j \rightarrow T_k$. T_j is said to be the ancestor of T_k , and T_k is said to be the descendant of T_j . Note that $T_j \rightarrow T_k \subset T_j \Rightarrow T_k$.

We are also given a linear ordering $L: (T_{k_1}, T_{k_2} \dots T_{k_n})$ of T , containing all T_k , called a Task List. When a P_i completes a task, it scans L from the beginning until it comes to the first executable task which has not yet begun to be executed, and immediately commences to work on that task. The whole scanning process is assumed to take no time. If P_i finds no such task, it becomes idle until some other processor P_j completes a task. At that time, all idle processors (including P_j) scan the list for tasks to execute. If two processors, P_i and P_j , $i < j$, simultaneously try to begin a task T_k , our convention will be to assign T_k to P_i , the processor with the smallest index.

The processors all start scanning at $t = 0$, and continue as above until all tasks have been executed.

The activity of the P_i is conveniently represented by a Gantt Chart⁴ consisting of n horizontal half-lines, each representing a processor. Time is measured horizontally and proceeds from the half-lines' common origin ($t = 0$) rightward. P_i is represented by the i th half-line from the top.

If processor P_i picks up a task T_k at time t_k^s and finishes it at time t_k^f , then the i th half-line is imagined to be "engaged" by T_k from t_k^s to t_k^f .

The Gantt chart shows the assignment of each task to a single processor.

Task run times t_k vary from zero to $t_{k(max)}$; our discussion concerns any run obeying these constraints.

Def. 1. A Standard Gantt Chart (SGC) is a Gantt Chart that is obtained by allowing all the t_k to equal $t_{k(max)}$. Given L and a specific partial ordering of the T_k , the SGC is determined, since there is in this case no variability of any parameter.

Def. 2. A Projective Task List (PL) is an ordered list of tasks whose order is the time-order in which tasks are picked up on the SGC. If two tasks are picked up at the same time, processor indexing determines which of the two comes first on the PL; the task picked up by the lower-indexed processor comes first on the PL.

Lemma 1. Given n , a partial ordering $\sqsubset$, and $t_k = t_{k(max)}$, the Gantt chart will be unchanged when the PL is substituted for L .

To prove this, we first give

Def. 3. Let T_j be defined as occupying the j th position on the PL.

Then by $T_{<j}$ we mean the set of tasks occurring before T_j on the PL; by $T_{>j}$ the set of tasks occurring after. The definition of $T_{\geq j}$ and $T_{\leq j}$ is the obvious one.

The proof of L. 1 is by induction. Suppose the $T_{<j}$ are mapped onto the n processors in exactly the position occupied by them on the SGC. Now after the $T_{<j}$ have been picked up, the next occasion for picking up a task must be at the place on the SGC where T_j is picked up. Furthermore, T_j is executable, else it could not have been picked up on the SGC, and it is the topmost unstarted task on the PL by definition. Therefore T_j will occupy the same position as it did on the SGC. Now T_1 , the topmost task on the PL, is obviously mapped onto the Gantt chart in its SGC position, therefore so is T_2 , etc. This proves Lemma 1.

Def. 4.1. In a typical run, T_k , n , and Γ are specified, but t_k can assume any value between 0 and $t_{k(\max)}$, where $t_{k(\max)}$ is known for all k . L is required to be the PL. By a "Gantt Chart" we mean a map showing the activity of the processors during a typical run.

Def. 4.2 By T_{-j} we mean the set of tasks picked up before T_j on the Gantt chart. If two tasks are picked up at the same time, we refer to the one picked up by the lower-indexed processor as being picked up "before" the one picked up by the higher-indexed processor.

Our discussion of Richards' Anomaly in Sec. I establishes that the next definition is not void of content:

Def. 5. A task T_k is said to be on time if its start time t_k^s is equal to or less than its start time on the SGC, which we write $t_{k(\text{SGC})}^s$. A set of tasks is said to be on time if all its component tasks are on time.

Corollary. If T_k is on time, then it ends on time, e.g.,

$t_k^f \leq t_k^f(\text{SGC})$. The proof is straightforward: $t_k^s \leq t_k^s(\text{SGC})$ (def. 5); $t_k^f \leq t_k^f(\text{SGC})$ (hyp.); therefore since $t_k^f = t_k^s + t_k^f$, $t_k^f \leq t_k^f(\text{SGC})$. Q.E.D.

Lemma 2. If $T_{-j} \cap T_{>j} = \Lambda$ [the null set] and T_{-j} are all on time, then T_j will be on time.

Proof. Suppose $t_j^s > t_j^s(\text{SGC})$. It is easy to demonstrate from first principles that every task is either picked up at $t = 0$ or else immediately after some other task terminates. Suppose T_j is picked up when T_x terminates. Clearly $T_x \in T_{-j}$. But $T_{-j} \cap T_{>j} = \Lambda$, therefore $T_{-j} \subset T_{<j}$ and $T_x \in T_{<j}$. In order to end on time, T_x must be a member of a subset of $T_{<j}$, $T_{<j}^2$, defined by $\forall r \in T_{<j}^2$, $t_r^f(\text{SGC}) > t_j^s(\text{SGC})$. But $t_r^s(\text{SGC}) < t_j^s(\text{SGC})$, therefore, on the SGC, the tasks in $T_{<j}^2$ occupy at most one processor each, and do not occupy the processor that executes T_j . Hence there are at most $n-1$ tasks in $T_{<j}^2$. Moreover, in the interval $\tau: (t_j^s(\text{SGC}), t_j^s)$ no task from $T_{<j}$ not an element of $T_{<j}^2$ can be running (else all T_{-j} will not be on time) and no task from $T_{>j}$ can be running (else $T_{-j} \cap T_{>j} \neq \Lambda$). We therefore conclude that there exists a non-empty set of P' of processors that are idle throughout τ . This, of course, implies that no tasks are executable for a finite amount of time during τ . Consequently T_j must suddenly become executable when T_x terminates. Therefore $T_x \rightarrow T_j$. But $t_x^f(\text{SGC}) > t_j^s(\text{SGC})$; therefore $T_x \not\rightarrow T_j$. We thus have a contradiction. Hence, $t_j^s \leq t_j^s(\text{SGC})$.

Lemma 3. Let $T_{k_1}, T_{k_2} \dots T_{k_r}$ be a sequence of tasks whose order is in the order in which they are picked up during a typical run. Let T_{k_1}

satisfy the hypotheses of lemma 2, i.e., $T_{>k_1} \cap T_{-k_1} = \Lambda$ and all T_{-k_1} are on time. Then if the sequence $k_1, k_2 \dots k_r$ is monotone increasing, all the $T_{k_1} \dots T_{k_r}$ will be on time.

Proof. By lemma 2, T_{k_1} is on time. By lemma 2 again, T_{k_2} is also on time, since by hypothesis $T_{k_1} \in T_{<k_2}$; and therefore it is true for $T_{k_1} \dots T_{k_r}$ by iteration.

Corollary. If all tasks are picked up in their order of appearance on the PL, they will all be on time.

Remark. The following two conjectures, which look like "generalizations" of lemma 2, are false. 1) If for some j , $T_{-j} \cap T_{>j} = \Lambda$, then T_j will be on time. 2) If for some j , T_{-j} are all on time, then T_j will be on time.

The disproof is by simple example and is left as an exercise for the reader.

Def. 7. A subset of tasks of order z is a collection of tasks T_{k_p} , $p = 1, 2, \dots, s$ in which there is a subset T_{k_r} , $r = 1, 2, \dots, z$, such that for all p , there exists some r such that either $T_{k_r} \Rightarrow T_{k_p}$ or $T_{k_r} = T_{k_p}$, and such that no other T_{k_r} with fewer than z members can be found in T_{k_p} that will satisfy these relations.

Remarks. Informally, Def. 7 says that the subset T_{k_p} has z roots.

The null set of tasks Λ is said to be a task subset of order zero.

Def. 8. A subset $\mathcal{G}^q$ consists of all the unfinished tasks in $T_{<q}$ when T_q is picked up during a typical run. $\mathcal{G}^q$ is then said to be defined by T_q . By $\mathcal{G}_i^q$ is meant the i^{th} task in $\mathcal{G}^q$ on the PL. By $\mathcal{G}_{<i}^q$ is meant the tasks in $\mathcal{G}^q$ occurring before $\mathcal{G}_i^q$ on the PL.

Lemma 4. At the moment that some task T_q is picked up, the unfinished tasks in $T_{<q}$, $\mathcal{G}^q$, form a subset of order z , where $z \leq n-v-1$; v is the number of tasks in $T_{>q}$ that are running when T_q is picked up. (If a task T_k starts at the same time as T_q , it is said to be running when T_q is picked up if and only if it is picked up by a lower-indexed processor.)

Remark: For the SGC, $v = 0$.

Proof. At any instant of time, any task either (1) is executable, (2) is running, (3) is already completed, or (4) has a parent task that has not yet been completed (tasks having no ancestors cannot be in category (4).) These categories are exhaustive and disjoint.

Now when T_q is picked up, no task in $T_{<q}$ can be executable. Hence at t_q^s , all tasks in $\mathcal{G}^q$ are in categories (2) and (4) above. If some $T_s \in \mathcal{G}^q$ is in category (4), then at least one of its parents, T_p , must be in categories (1), (2), or (4). Category (1) can be ruled out, however, since $T_s \in T_{<q} \subset T_p \in T_{<q}$; hence T_p is not executable. By induction we can work back as far as an initial task which has no ancestor and cannot be in category (4). Hence we conclude that at t_q^s all tasks in $\mathcal{G}^q$ are either running or have ancestors that are running. Clearly, no ancestor of a task in $\mathcal{G}^q$ that is running at t_q^s can be in $\mathcal{G}^q$ [cf. def. 8]. Therefore the tasks in $\mathcal{G}^q$ running at t_q^s are precisely the z "roots" T_{k_r} [cf. def. 7] of $\mathcal{G}^q$. Hence at t_q^s , only $n - z - 1$ processors are available to tasks in $T_{>q}$. Therefore $v \leq n - z - 1$, and $z \leq n - v - 1$.

Lemma 5. At all times up to the time any task T_j is picked up, some task in $T_{<j}$ is always running.

Remark. This lemma is not required for our theorem, but it seems to be worth inclusion. Only a sketch of the proof is given.

Proof. If the hypothesis of Lemma 5 were untrue, then there would be some time before t_j^s at which all processors were either executing tasks from $T_{>j}$ or else were idle.

Now it can be shown easily that the following statement is true: If T_j has not been picked up, then at any time $t < t_j^s$, either T_j is executable or there exists some $T_m \Rightarrow T_j$ that is either running or executable. If all processors are idle or busy with tasks from $T_{>j}$ at t , it means that there must have been at least one time $t' \leq t$ when no tasks from $T_{<j}$ were either running or executable. But this cannot happen, since $T_m \in T_{<j}$.

Lemma 6. Suppose a typical run is conducted with n' processors, where n' always remains $\geq n$, the number of processors in the SGC. Then lemmas 2 and 3 are still valid.

Proof. The argument justifying lemma 1 deals only with the SGC in discussion timing relations, as can be verified by inspection. However, the statement in lemma 2 that the limitation on the number of tasks in $T_{<j}^2$ implies an idle processor subset P' is obviously true in general when $n' \geq n$. Lemma 3 then follows as before from lemma 2. Q.E.D.

Remark. Since Graham's theorem gives in general $\omega'/\omega > 1$ even when $n < n'$, lemma 6 is not devoid of content.

Remark. Lemma 6 says that the use of a PL eliminates Richards' Anomaly of type 3 (see Section I) for $n' > n$. Furthermore increasing

the number of processors never causes an increase in the start time of any task.

Lemma 7. If for all q , no subset $\mathcal{C}^q$ includes within it any three tasks T_α , T_β , and T_γ , such that A) $T_\gamma \Rightarrow T_\alpha$, B) $T_\gamma \Rightarrow T_\beta$, and C) T_α and T_β have finite timing overlap* on the SGC, then all tasks will be completed on time.

Proof. Let $T_{j_1}, T_{j_2}, \dots, T_{j_m}, \dots$ be a sequence of tasks given in the order in which they are picked up during a typical run. We now show that the hypotheses imply that if T_{j_m} is on time, then T_{j_m} will be on time; therefore by induction all tasks will be on time. (1) If

$T_{j_m} \cap T_{j_m} = \Lambda$, T_{j_m} will be on time by lemma 2. (2) Suppose therefore $T_{j_m} \cap T_{j_m} \neq \Lambda$. Let $T_{q(j_m)}$ be defined as the task in $\{T_{j_m} \cap T_{j_m}\}$ with the highest value of i . Then $T_{q(j_m)}$ is simply the task in T_{j_m} that is the last to be picked up before T_{j_m} is picked up. Therefore from the time $T_{q(j_m)}$ is picked up to the time T_{j_m} is picked up, every task that is picked up is (A) in T_{j_m} and (B) in $\mathcal{C}^{q(j_m)}$, where $\mathcal{C}^{q(j_m)}$ is defined [def. 8] by $T_{q(j_m)}$.

Now when $T_{q(j_m)}$ is picked up, the z "root" tasks [def. 7] of $\mathcal{C}^{q(j_m)}$ are running (discussion, lemma 4) and all of these tasks are in T_{j_m} . In accordance with our induction, we may suppose them to be on time.

If the hypotheses of Lemma 7 are satisfied, it follows with a little reflection that the tasks in any subset $\mathcal{C}^q$ are so arranged on the SGC that at most z of them are ever running concurrently, where z is the order of $\mathcal{C}^q$. Consequently, these tasks may be regarded as mapped

* (See p. III F-42, bottom)

onto a SGC with z processors.

Since the z root tasks occupy z processors at $t_{q(j_m)}^s$, it follows from (B) of the last line of (2) above that at least z processors remain available for executing $\delta^{q(j_m)}$ up to and including the time T_{j_m} is picked up. The definition of $T_{q(j_m)}$ implies $\delta_{>j_m}^{q(j_m)} \cap \delta_{-j_m}^{q(j_m)} = \Lambda$, and all $\delta_{-j_m}^{q(j_m)}$ may be supposed on time, since $\delta_{-j_m}^{q(j_m)} \subset T_{-j_m}$. By lemma 6, it follows that T_{j_m} is on time, since the system $\delta_{-j_m}^{q(j_m)}$ can be regarded without loss of generality as a task set mapped onto a SGC with z processors, for which task set there are at least z processors available during a typical run.

Lemma 8. Our "stability algorithm" implies the satisfaction of the hypotheses of lemma 7, and thereby assures the results of the stability theorem.

We first prove the following two trivial lemmas:

Lemma 9: If an arbitrary vertical line on the SGC divides all tasks into "Left" and "Right," [see (4) of the stability algorithm] then any "right" task either has parents all of which are "left" or else has at least one "right" ancestor all of whose parents are "left."

Proof. Trace back along the task set, following successive "right" parents. Since each parent is earlier than its son, a parent must be found whose parents are all "left." Q.E.D.

* That is, either $t_{\alpha}^s(\text{SGC}) < t_{\beta}^s(\text{SGC})$ and $t_{\alpha}^f(\text{SGC}) > t_{\beta}^f(\text{SGC})$ or

$$t_{\alpha}^f(\text{SGC}) > t_{\beta}^s(\text{SGC}) \text{ and } t_{\beta}^f(\text{SGC}) > t_{\alpha}^s(\text{SGC})$$

Lemma 10. The stability algorithm guarantees that every "right" task T_R will enter into a precedence relation of the form $T_A \Rightarrow T_R$.

Proof. Either it was originally the case that $T_A \Rightarrow T_R$, or else by the stability algorithm a stability relation $T_A \rightarrow T'_R$ will be constructed, where by Lemma 9, $T'_R \Rightarrow T_R$. Hence $T_A \Rightarrow T_R$. Q.E.D.

We now prove lemma 8. Let three tasks T_α , T_β , T_γ , be related by $T_\gamma \Rightarrow T_\alpha$ and $T_\gamma \Rightarrow T_\beta$ and let T_α and T_β have finite timing overlap on the SGC. Suppose some subset σ^q has the property that $T_\alpha \in \sigma^q$ and $T_\beta \in \sigma^q$. Then from def. 8, it follows that if σ^q is defined by T_q , then $T_q \in T_{>\alpha}$ and $T_q \in T_{>\beta}$. Now by step (4) of the stability algorithm, either T_α or T_β or some task starting earlier than both will have a starting time that will divide the SGC into a "Right" and "Left" region. Clearly T_q is a "right" task. By lemma 10, the nearest common ancestor of T_α and T_β , (call it T_A), will have property that $T_A \Rightarrow T_q$ when the stability conditions are added. But from the definition of nearest common ancestor [(3), stability algorithm] either $T_\gamma = T_A$ or $T_\gamma \Rightarrow T_A$. Consequently, $T_\gamma \Rightarrow T_q$. Since this implies that T_γ must be finished at the time T_q is picked up, it follows that a subset containing T_α and T_β cannot contain T_γ . Q.E.D.

(b) Superfluity of Some Stability Conditions.

The stability algorithm yields additional precedence relations which are sufficient to stabilize the task set. Some of these conditions are not, it turns out, actually necessary. Nor is their choice unique. It is hard to say what one's criteria ought to be in seeking a set of stability conditions that is somehow "minimal". Should one look for the minimum number of added conditions? Or should one attempt to find a set of conditions that will somehow cause the least delay during an "average" typical run? Or is there some "better criterion?"

The writer has no idea of how to solve any of these problems. Suffice to say that in certain special situations, some of the stability conditions derived via our stability algorithm are believed to be superfluous. In the one case discussed below, we lack a proof, but simulation studies support this conjecture:

Conj. On the SGC, let the time-gap between the end of an ancestor task T_A [cf. (3) of the stability algorithm] to the beginning of the earliest-starting member of any pair whose common ancestor is T_A [cf. (4), *ibid*] be ϵ_A . Then $T_A \rightarrow T_R$ [cf. (5), *ibid*] is necessary only if either $t_{R(\max)} > \epsilon_A$ or if there exists some task T_Q such that $T_R \Rightarrow T_Q$ and $t_{Q(\max)} > \epsilon_A$.

The significance of this conjecture, if true, has to do with our notion of "granularity". Up to now, we have supposed that the $t_{j(\max)}$ are arbitrary. Suppose, however, that we impose a limit: we require that for all j , $t_{j(\max)} \leq M$. Then if for some T_A , $\epsilon_A \geq M$, $T_A \rightarrow T_R$

will be superfluous. (One might suppose that one could then avoid stability conditions altogether by constructing SGC's that always have

$\epsilon_A \geq M$ for all T_A . An experimental program written by Miss V. Bishop of Bell Telephone Laboratories has confirmed, however, that this is a very inefficient strategy.)

APPENDIX D

Guidelines for a possible proof of the "Time-Sharing" Procedure.

We assume we are dealing with projective lists and that the stability algorithm is followed. Let T_k be the first task in the PL whose F field is down. At the time t that we are preparing to "time-share" a processor, a sequence of tasks $T_{j_1} \dots T_{j_m}$ has been picked up and all of them are on time. At t a processor becomes free.

By the definition of T_k , all the tasks that have not been picked up are in $T_{\leq k}$, and therefore have starting times on the SGC $\geq t_{k(SGC)}^s$, the contents of the E field of T_k . Let $\delta \equiv t_{k(SGC)}^s - t$. Suppose δ happens to be positive. During the time the processor is removed from the job - provided only that it is not removed for a time greater than δ - all tasks must be on time. Following the return of the processor to the system, the system will "look" as though some sequence $T_{j_1} \dots T_{j_m} \dots T_{j_{m+1}} \dots T_{j_{m+n}}$ had been executed all on time.

Hence we conjecture that a proof can be found showing that all tasks will be on time.

REFERENCES

1. BIRKHOFF, G., Lattice Theory. A.M.S. Coll. Publications, Vol. XXV, (1940).
2. RICHARDS, P. Timing Properties of Multiprocessor Systems Tech. Ops. Inc. Tech. Paper, Report No. TD-B60-27, August 1960.
3. ROY, B. Les Problèmes d'Ordonnancement, Monographie de Recherche Operationnelle No. 2, Dunod, Paris, 1964. This monograph discusses relative and absolute timing constraints in considerable detail.
4. HELLER, J. Sequencing Aspects of Multiprocessing. JACM, 1961, vol 8, p. 426-439.
5. HELD, Michael, and KARP, Richard M., A Dynamic Programming Approach to Sequencing Problems, JSIAM, 10, 1, March, 1962. This article includes an excellent bibliography.
6. GRAHAM, R.L., paper on boundary conditions for timing in task sets (to be published).
7. WIEST, Jerome D., Some Properties of Schedules for Large Projects with Limited Resources, Operations Research 12, 3, May-June 1964.
8. FARBER, D.J., GRISWOLD, R.E., POLONKY, I.P., SNØBØL, A String Manipulation Language. JACM 11, 1, 1964.
9. McNAUGHTON, R. Scheduling with Deadlines and Loss Functions - Management Science, 6, No. 1, Oct. 1954, pp. 1-12.
10. CODD, E.L., Multiprogram Scheduling, JACM, June 1960, vol. 3, no. 6, pp. 347-350.
11. OCHSNER, B.P., Controlling a multiprocessor system, Bell Laboratories Record, February, 1966.

FIG. 1

FIG. 2

FIG. 3

FIG. 4

FIG. 5

FIG. 6

FIG. 7

FIG. 8

FIG. 9

FIG. 10

FIG. 11

FIG. 12

FIG. 13

FIG. 14

FIG. 17

FIG. 18

FIG. 19

FIG. 20

FIG. 21

FIG. 22

"ROADMAP"
TASK MODEL

FIG. 23

FIG. 24

FIG. 25

TASK LENGTHS = $.67 \pm .33$

FIG. 26

FIG. 27

FIG. 28

FIG. 29

FIG. 30

FIG. 31

FIG. 32