3. 行列計算の自動プログラミング

島内武彦(東大理)

行列計算のプログラミングは手間がかかるのでこれを簡単に行いたい。たとえば

$A+B=C.$

$A+B+C=D.$

$A\times B=C.$

$A\times B\times C=D.$

$(A\times B)+C=D.$

などの計算を普通の数の計算と同じプログラミングで行えるようにしたい。

それにはこれが行えるインタープリーテイブ・サブルーチンあるいはこのための自動プログラミング用ルーチンをつくればよい。その準備としてまず行列計算のプログラムを出来るだけsystematic にすることが必要である。それについて述べる。

行列計算の体系化の一案

これはつぎのような5つのプロセスから成る。

(1) 行列の数値の store の方法

(2) 準備ルチーン

(3) 計算ルチーン

(4) イコール・ルチーン

(5) ピリオド・ルチーン

$A+B=C$ を例としてこれらを説明する。

(1) 行列の数値の store の方法

これは各要素の数値を順に入れる前に次のような8語の余分の space

をつくる。

	A 行列	B 行列
行の数	m_A	m_B
列の数	n_A	n_B
行を進める時にアドレスに加える数	$\varDelta m_A$	$\varDelta m_B$
列を進める時にアドレスに加える数	$\varDelta n_A$	$\varDelta n_B$
Working space	α_1	β_1
〃	α_2	β_2
〃	α_3	β_3
最初の要素のアドレス	α_4	β_4
要素の数値を順に入れる	…	…
(縦の順でも横の順でもよい)	…	…
	…	…

(2) 準備ルチーン

(2a) C 行列を入れるべきアドレスの処に順に次の数を set する。

行の数 ($m_C = m_A$), 列の数 ($n_C = n_B$), $\varDelta m_C$, $\varDelta n_C$

次に Working space 3語 (γ_1, γ_2, γ_3) を reserve し,

次に C 行列の最初の要素のアドレスを γ_4 として set する。

(2b) 行列 Working space に各行列の最初の要素のアドレスを set する。

α_1, α_2, α_3 に α_4 を, β_1, β_2, β_3 に β_4 を,

γ_1, γ_2, γ_3 に γ_4 を set する。

(2c) Working space 0 を clear する。

(3) 計算ルチーン

各行列の名前の代りに α_1, β_1, γ_1 などを使つて普通の計算のルチーンをつくる。ただし=のところはイコール・ルチーンに立寄らせ, 式の終りのピリオドのところではピリオド・ルチーンに立寄らせる。

たとえばPC-1 の code を使えば次のようにする。

$p l \alpha_1$	α_1 番地にある数をとり出す。
$v l \beta_1$	これに β_1 番地の数をかける。
$a l 0$	これに 0 番地の数を加える。
$t l 0$	これを 0 番地に入れる。
$i t$	} イコール・ルチーンに立寄る。
$j l$ (イコール・ルチーンのアドレス)	
$p l 0$	0 番地の数をとり出す。
$t l \gamma_1$	これを γ_1 番地に入れる。
$i t$	} ピリオド・ルチーンに立寄る。
$j l$ (ピリオド・ルチーンのアドレス)	
$w l t.$	STOP

(4) イコール・ルチーン

答を出す前に A, B の要素を交代させて繰り返し計算を行わせるサブルチーンである。

$A \times B = C$ の場合は次のようになる。

(4a) α_1 に Δn_A を加える。

(4b) β_1 に Δm_B を加える。

(4c) 計算ルチーンのはじめにもどる。

(4d) これを n_A 回くり返したら先に進む。

問題は計算式がちがつた時にこのルチーンがどう代るかということである。これについては表1に示す。

(5) ピリオド・ルチーン

答を一つ出してから更に A, B の要素を交代させて順次に答を出させるルチーンである。$A \times B = C$ の時には次のようになる。

(5a) 0 を clear

(5b) α_2 に Δm_A を加える。

(5c) α_2 を α_1 に入れる

(5d) β_2 を β_1 に入れる

(5e) γ_1 に Δm_C を加える

(5f) 計算ルチーンのはじめにもどる

(5g) 以上を m_C 回くり返したら先に進む

(5h) α_3 を α_2 に入れる

(5i) α_2 を α_1 に入れる

(5j) β_2 に Δn_B を加える

(5k) β_2 を β_1 に入れる

(5l) γ_2 に Δn_C を加える

(5m) γ_2 を γ_1 に入れる

(5n) 計算ルチーンのはじめにもどる

(5o) 以上を n_C 回くり返す

計算式のちがいによるルチーンのちがいは表1に示す。

注意(1) この方法によれば行列の要素の数値はつづけて入れなくてもとびとびに入れることも可能である。行列の一部分のみを小行列またはベクトルとして取り扱える。

(2) 対称行列の場合には三角形の行列として store するが，この場合には Δm, Δn を順次に変化させる。

(3) 計算式のちがいによるイコール・ルチーン，ピリオド・ルチーンの変化を表1に示す。表では α_3 は Working space として用いていないが，これは更に複雑な計算，または対称行列の計算に working space として利用できる。

(4) $A'BA=C$ のように同じ行列が2回以上出てくる時には8語の space だけを計算に出る回数だけつくっておく。転置行列の場合には m と n，Δm と Δn とを逆にしたものをつくっておく。

(5) *$A\times B\times C\times D$ のように乗法を連続して行う場合には表1のルチーンでは同一計算の繰返しが多く不経済なので，プログラムを少し変えてこれを避けるか，または

$$A\times B=M$$

$$M\times C=N$$

$$N\times D=E$$

のように一回の乗法毎に答をつくるように式を変更する。計算の途中で出てくる M，N 等は他の行列とかなり重ねて store してよい。

表 1.

$A+B=C$.	$A+B+C=D$.	$A\times B=C$.		$A\times B\times C=D$.	$A\times B\times C\times D=E$.	$(A\times B)+C=D$.
(イコール) なし	(イコール) なし	(イコール) $\begin{cases}\alpha_1+\Delta n_A\rightarrow\alpha_1\\ \beta_1+\Delta m_B\rightarrow\beta_1\end{cases}$ 以上をn_A回くり返す		(イコール) $\begin{cases}\alpha_1+\Delta n_A\rightarrow\alpha_1\\ \beta_1+\Delta m_B\rightarrow\beta_1\end{cases}$ 以上をn_A回くり返す $\begin{cases}\alpha_2\rightarrow\alpha_1\\ \beta_2+\Delta n_B\rightarrow\beta_2\rightarrow\beta_1\\ \gamma_1+\Delta m_C\rightarrow\gamma_1\end{cases}$ 以上全部をn_B回くり返す	(イコール) $\begin{cases}\alpha_1+\Delta n_A\rightarrow\alpha_1\\ \beta_1+\Delta m_B\rightarrow\beta_1\end{cases}$ 以上をn_A回くり返す $\begin{cases}\alpha_2\rightarrow\alpha_1\\ \beta_2+\Delta n_B\rightarrow\beta_2\rightarrow\beta_1\\ \gamma_1+\Delta m_C\rightarrow\gamma_1\end{cases}$ 以上全部をn_B回くり返す $\begin{cases}\beta_3\rightarrow\beta_2\rightarrow\beta_1\\ \gamma_2+\Delta n_C\rightarrow\gamma_2\rightarrow\gamma_1\\ \delta_1+\Delta m_D\rightarrow\delta_1\end{cases}$ 以上全部をn_C回くり返す	(カツコ・ルチーン)* $\begin{cases}\alpha_1+\Delta n_A\rightarrow\alpha_1\\ \beta_1+\Delta m_B\rightarrow\beta_1\end{cases}$ 以上をn_A回くり返す (イコール) なし
(ピリオド) O を clear $\begin{cases}\alpha_1+\Delta n_A\rightarrow\alpha_1\\ \beta_1+\Delta n_B\rightarrow\beta_1\\ \gamma_1+\Delta n_C\rightarrow\gamma_1\end{cases}$ 以上をn_C回くり返す $\begin{cases}\alpha_2+\Delta m_A\rightarrow\alpha_2\rightarrow\alpha_1\\ \beta_2+\Delta m_B\rightarrow\beta_2\rightarrow\beta_1\\ \gamma_2+\Delta m_C\rightarrow\gamma_2\rightarrow\gamma_1\end{cases}$ 以上全部をm_C回くり返す	(ピリオド) O を clear $\begin{cases}\alpha_1+\Delta n_A\rightarrow\alpha_1\\ \beta_1+\Delta n_B\rightarrow\beta_1\\ \gamma_1+\Delta n_C\rightarrow\gamma_1\\ \delta_1+\Delta n_D\rightarrow\delta_1\end{cases}$ 以上をn_D回くり返す $\begin{cases}\alpha_2+\Delta m_A\rightarrow\alpha_2\rightarrow\alpha_1\\ \beta_2+\Delta m_B\rightarrow\beta_2\rightarrow\beta_2\\ \gamma_2+\Delta m_C\rightarrow\gamma_2\rightarrow\gamma_1\\ \delta_2+\Delta m_D\rightarrow\delta_2\rightarrow\delta_1\end{cases}$ 以上全部をm_D回くり返す	(ピリオド)　または O を clear $\begin{cases}\alpha_2+m_A\rightarrow\alpha_2\rightarrow\alpha_1\\ \beta_2\rightarrow\beta_1\\ \gamma_1+\Delta m_C\rightarrow\gamma_1\end{cases}$ 以上をm_C回くり返す $\begin{cases}\alpha_3\rightarrow\alpha_2\rightarrow\alpha_1\\ \beta_2+\Delta n_B\rightarrow\beta_2\rightarrow\beta_1\\ \gamma_2+\Delta n_C\rightarrow\gamma_2\rightarrow\gamma_1\end{cases}$ 以上全部をn_C回くり返す	(ピリオド) O を clear $\begin{cases}\alpha_2\rightarrow\alpha_1\\ \beta_2+\Delta m_B\rightarrow\beta_2\rightarrow\beta_2\\ \gamma_1+\Delta n_C\rightarrow\gamma_1\end{cases}$ 以上をn_C回くり返す $\begin{cases}\alpha_2+\Delta m_A\rightarrow\alpha_2\rightarrow\alpha_1\\ \beta_3\rightarrow\beta_2\rightarrow\beta_1\\ \gamma_2+\Delta m_C\rightarrow\gamma_2\rightarrow\gamma_1\end{cases}$ 以上全部をm_C回くり返す	(ピリオド) O を clear $\begin{cases}\beta_3\rightarrow\beta_2\rightarrow\beta_1\\ \gamma_2+\Delta n_C\rightarrow\gamma_2\rightarrow\gamma_1\\ \delta_1+\Delta n_D\rightarrow\delta_1\end{cases}$ 以上をn_D回くり返す $\begin{cases}\alpha_2+\Delta m_A\rightarrow\alpha_2\rightarrow\alpha_1\\ \gamma_3\rightarrow\gamma_2\rightarrow\gamma_1\\ \delta_2+\Delta m_D\rightarrow\delta_2\rightarrow\delta_1\end{cases}$ 以上全部をm_D回くり返す	(ピリオド) O を clear $\begin{cases}\gamma_3\rightarrow\gamma_2\rightarrow\gamma_1\\ \delta_2+\Delta n_D\rightarrow\delta_2\rightarrow\delta_1\\ \varepsilon_1+\Delta n_E\rightarrow\varepsilon_1\end{cases}$ 以上をn_E回くり返す $\begin{cases}\alpha_2+\Delta m_A\rightarrow\alpha_2\rightarrow\alpha_1\\ \delta_3\rightarrow\delta_2\rightarrow\delta_1\\ \varepsilon_2+\Delta m_E\rightarrow\varepsilon_2\rightarrow\varepsilon_1\end{cases}$ 以上全部をm_E回くり返す	(ピリオド) $\begin{cases}\alpha_2\rightarrow\alpha_1\\ \beta_2+\Delta n_B\rightarrow\beta_2\rightarrow\beta_1\\ \gamma_1+\Delta n_C\rightarrow\gamma_1\\ \delta_1+\Delta n_D\rightarrow\delta_1\end{cases}$ 以上をn_D回くり返す $\begin{cases}\alpha_2+\Delta m_A\rightarrow\alpha_2\rightarrow\alpha_1\\ \beta_3\rightarrow\beta_2\rightarrow\beta_1\\ \gamma_2+\Delta m_C\rightarrow\gamma_2\rightarrow\gamma_1\\ \delta_2+\Delta m_D\rightarrow\delta_2\rightarrow\delta_1\end{cases}$ 以上全部をm_D回くり返す (* カツコ・ルチーンは計算ルチーンのカツコの場所にサブルチーンとして挿入する。)

本 PDF ファイルは 1960 年発行の「第 1 回プログラミング–シンポジウム報告集」をスキャンし、項目ごとに整理して、情報処理学会電子図書館「情報学広場」に掲載するものです。

この出版物は情報処理学会への著作権譲渡がなされていませんが、情報処理学会公式 Web サイトの `https://www.ipsj.or.jp/topics/Past_reports.html` に下記「過去のプログラミング・シンポジウム報告集の利用許諾について」を掲載して、権利者の捜索をおこないました。そのうえで同意をいただいたもの、お申し出のなかったものを掲載しています。

過去のプログラミング・シンポジウム報告集の利用許諾について

情報処理学会発行の出版物著作権は平成 12 年から情報処理学会著作権規程に従い、学会に帰属することになっています。

プログラミング・シンポジウムの報告集は、情報処理学会と設立の事情が異なるため、この改訂がシンポジウム内部で徹底しておらず、情報処理学会の他の出版物が情報学広場 (=情報処理学会電子図書館) で公開されているにも拘らず、古い報告集には公開されていないものが少からずありました。

プログラミング・シンポジウムは昭和 59 年に情報処理学会の一部門になりましたが、それ以前の報告集も含め、この度学会の他の出版物と同様の扱いにしたいと考えます。過去のすべての報告集の論文について、著作権者（論文を執筆された故人の相続人）を探し出して利用許諾に関する同意を頂くことは困難ですので、一定期間の権利者捜索の努力をしたうえで、著作権者が見つからない場合も論文を情報学広場に掲載させていただきたいと思います。その後、著作権者が発見され、情報学広場への掲載の継続に同意が得られなかった場合には、当該論文については、掲載を停止致します。

この措置にご意見のある方は、プログラミング・シンポジウムの辻尚史運営委員長 (`tsuji@math.s.chiba-u.ac.jp`) までお申し出ください。

加えて、著作権者について情報をお持ちの方は事務局まで情報をお寄せくださいますようお願い申し上げます。

期間：2020 年 12 月 18 日 ～ 2021 年 3 月 19 日
掲載日：2020 年 12 月 18 日

プログラミング・シンポジウム委員会

情報処理学会著作権規程
`https://www.ipsj.or.jp/copyright/ronbun/copyright.html`