الگوریتم‌های موازی بهینه در حل دستگاه‌های تاپلیتز بر روی شبکه‌های مش و فوق مکعبی

داود رستمی، حسن علیزاده
دانشکده ریاضی، دانشگاه بین‌المللی امام خمینی قزوین، ایران
* مسئول مکاتبات- آدرس الکترونیکی: *Rostamy@Khayamut.ac.ir*
(دریافت: ۸۴/۲/۲۹ ؛ پذیرش:۸۶/۱/۲۵)

چکیده

در این مقاله برای اولین بار پیاده سازی واجرای الگوریتم های موازی بر روی شبکه های مش (Mesh) و فوق مکعبی (Hypercube) برای حل سیستم های خطی تاپلیتز توسط روش Preconditioned Conjugate Gradient (PCG) ارا ئه گردیده است. ارزش تمام الگوریتم های ارائه شده محا سبه و بهینه بودن آن اثبا ت می گردد. همچنین اجرای الگوریتم ها ی ارائه شده در نرم افزار Parallel Virtual Machine (PVM) و محاسبه زمان اجرای تکرار و کارایی آنها با توجه به مثال‌های عددی برای ماتریس های تاپلیتز ارائه شده است.

واژه‌های کلیدی: محاسبات موازی، حل سیستم‌های خطی موازی، روش‌های زیر فضای کرایلف، ماتریس‌های تاپلیتز و شبکه‌های مش و فوق معکبی

مقدمه

امروزه کارهای بسیاری در حالت موازی سازی سیستم های خطی توسط نویسندهای متفاوتی در این خصوص انجام شده است (Chan *et al* 1989، Gupta *et al* 1992، Gupta *et al* 1992، Gallivan *et al* 1989، Misra et al. 1993، Miranker 1971، Qinn 1990، Sameh et al. 1977، Saad 2002، Saad 1988، Vagda 1993). این مقاله بر اساس ساختارهای جد ید فوق مکعبی برای ضرب ماتریس ها و با استفاده از تئوری جبر خطی عددی که برای ساختن ماتریس های ضربی مناسب برای حل این دستگاه‌ها است، ارائه شده است (Anisimov 1999، Bitmead & Anderson 1980، Bunch 1985، Davis 1979، Gupta et al. 1992، Strang. 1986). و این وجه تمایزعمده با سایر کارهای قبلی می باشد که پیچیدگی محاسبات را کاهش می دهد. در ضمن برای مقایسه روش مش (Berteskas et al., 1989، Saad, 2002) بعنوان شاخص در نظر گرفته شده است. بدین منتظور به مطالب زیر به عنوان مقدمه توجه می‌کنیم.

دستگاه خطی زیر که در آن $A_{n\times n}$ ماتریس حقیقی، متقارن و معین مثبت فرض می‌شود را در نظر بگیرید:

$$Ax = b \qquad (1)$$

روش PCG (Preconditioned Conjugate Gradient) برای حل این دستگاه بصورت ترتیبی توسط الگوریتم زیر بیان می‌شود. که در آن r_0, x_0 به ترتیب بردار اولیه و بردار باقیمانده می‌باشند. ماتریس $C_{n\times n}$ ماتریس پیش شرط نامیده می‌شود. که چگونگی محاسبه آن را با روشهای استرانگ و چن در ادامه بیان خواهیم کرد.

$\Pr ocedure \quad PCG(A, b, x_0, x)$
$Step1:1.1)\ ro \leftarrow b - Ax_0$
$1.2)\ Solve\, Cz_0 = r_0$
$1.3)\ p_0 \leftarrow z_0$
$Step2:\ for\, i = 0,1,2,...until\, convergence\, do$
$2.1)\ a_i \leftarrow (r_i, z_i)/(Ap_i, p_i)$
$2.2)\ x_{i+1} \leftarrow x_i + a_i p_i$
$2.3)\ r_{i+1} \leftarrow r_i - a_i Ap_i$
$2.4)\ Solve\, Cz_{i+1} = r_{i+1}$
$2.5)\ \beta_i \leftarrow (r_{i+1}, z_{i+1})/(r_i, z_i)$
$2.6)\ p_{i+1} \leftarrow z_{i+1} + \beta_i p_i$
$end\ for.$

توجه داریم که کران خطا برای روش PCG از رابطه زیر به دست می‌آید (Saad 2002).

$$\frac{\|e_k\|_A}{\|e_0\|_A} < 2\left(\frac{\sqrt{\lambda_{\max}} - \sqrt{\lambda_{\min}}}{\sqrt{\lambda_{\max}} + \sqrt{\lambda_{\min}}}\right)^k$$

که در آن $e_k = \|x^\circ - x_k\|_A$ بردار خطا در تکرار k ام و $\lambda_{\min}$ و $\lambda_{\max}$ به ترتیب کوچکترین و بزرگترین مقادیر ویژه ماتریس ضرائب و x° جواب دستگاه می‌باشد. این رابطه نشان می‌دهد هر چه مقدار $\frac{\lambda_{\max}}{\lambda_{\min}}$ کوچکتر باشد همگرایی سریعتر است. هچنین از این رابطه نتیجه می‌شود. نرخ همگرایی خطی است. بنابراین روش PCG بعد از تعداد

تکرارهای ثابتی به جواب واقعی می‌رسد. اما برای وقتی که مقدار $\frac{\lambda_{\max}}{\lambda_{\min}}$ بزرگ باشد تعداد تکرارها ممکن است زیاد باشد. در واقع تعداد تکرارهای روش PCG به مقادیر ویژه ماتریس ضرائب بستگی دارد. بنابراین انتخاب یک ماتریس پیش شرط مناسب میتواند باعث انباشته شدن مقادیر ویژه در اطراف عدد یک شده و در نتیجه با کوچک شدن مقدار $\frac{\lambda_{\max}}{\lambda_{\min}}$ سرعت همگرایی افزایش یابد. استرانگ و چن با قضایا و نتایج عددی نشان دادند ماتریسهای پیش شرطی از نوع ماتریسهای چرخشی وجود دارد که استفاده از آنها باعث افزایش قابل توجه سرعت همگرایی روش PCG می‌شود (Chan 1988، Chan *et al* 1987, 1989). بنابراین از این پس فرض می‌کنیم تعداد تکرارهای روش PCG مقداری ثابت است و به ابعاد ماتریس ضرائب بستگی ندارد.

تعریف ۱: ماتریس چرخشی ماتریس مرتبه n است که در آن هر سطر با انتقال به راست چرخشی از سطر قبلی بدست می‌آید. ماتریس چرخشی $C_{n\times n}$ را به صورت زیر نمایش می‌دهیم:

$$C_{n\times n} = circ(c_1, c_2, ..., c_n)$$

$$C_{n\times n} = \begin{bmatrix} c_1 & c_2 & \dots & c_{n-1} & c_n \\ c_n & c_1 & c_2 & \dots & c_{n-1} \\ \vdots & \vdots & & & \vdots \\ c_2 & c_3 & & & c_1 \end{bmatrix}$$

همانطور که مشاهده می‌شود عناصر هر سطر با سطرهای دیگر یکسان است تنها با این تفاوت که هر سطر نسبت به سطر قبلی، بصورت چرخشی یک مکان به سمت راست انتقال یافته است.

تعریف۲: ماتریس فوریه بصورت زیر تعریف می‌شود.

$$F = \frac{1}{\sqrt{n}} \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & w^{-1} & w^{-2} & \cdots & w^{-(n-1)} \\ 1 & w^{-2} & w-4 & \cdots & w^{-2(n-1)} \\ \vdots & \vdots & & & \vdots \\ 1 & w^{-(n-1)} & w^{-2(n-1)} & \cdots & w^{-(n-1)(n-1)} \end{bmatrix}_{n\times n}$$

تعریف۳: ماتریس فوریه ترانهاده مزدوج بصورت زیر تعریف می‌شود.

$$F^* = \frac{1}{\sqrt{n}}(w^{(i-1)(j-1)}), i, j = 1,2,...,n$$

F°, F متقارن و یکانی (Unitary) می‌باشند و داریم:

$$(\overline{F})^T = F^*$$

$$F = F^T, F^* = (F^*)^T = \overline{F}, F = \overline{F}^*$$

$$FF^* = F^*F = I \Leftrightarrow F^{-1} = F^* = (\overline{F})^T$$

قضیه ۱: قطری سازی ماتریسهای چرخشی.

۱) اگر C ماتریس چرخشی باشد آنگاه بوسیله F قابل قطری شدن است (Davis 1979).

۲) اگر $\Delta = diag(\lambda_0, \lambda_1, ..., \lambda_n)$ و λ_i مقادیر ویژه C باشند آنگاه $C = F^* \wedge F$ [۱۰].

نتیجه : اگر $C = Circ(c_0, c_1, ...c_{n-1})$ و λ_i ها مقادیر ویژه C باشند آنگاه (Davis 1979).

$$\lambda_j = c_o + c_1 w^j + c_2 w^{2j} + ... + c_{n-1} w^{(n-1)j} = \sum_{k=0}^{n-1} c_k \times w^{kj}, j = 0.1, ..., n-1$$

تعریف ۴: ماتریس مربعی $T = (t_{ij})$ را تاپلیتز (Toeplitz) گویند هرگاه: $(i, j = 1,2,...,n-1), t_{i,j} = t_{i+1,j+1}$

مثال ۱: ماتریس $T_{4\times4}$ تاپلیتز است.

$$T_{4\times4} = \begin{bmatrix} a & b & c & h \\ d & a & b & c \\ e & d & a & b \\ f & e & d & a \end{bmatrix}$$

بنابراین در حالت کلی ماتریس تاپلیتز $T_{n\times n}$ دارای حداکثر $(2n-1)$ عنصر مجزا بر روی قطرهای آن می‌باشد. همچنین از تعریف ماتریس چرخشی و تاپلیتز می‌توان به این نتیجه رسید که هر ماتریس چرخشی تاپلیتز است ولی هر ماتریس تاپلیتز چرخشی نیست.

تعریف: دستگاه $Ax = b$ را تاپلیتز گویند هر گاه A ماتریس تاپلیتز باشد.

۲- پیش شرطها

۲- ۱- معرفی پیش شرط استرانگ

پیش شرط استرانگ یک ماتریس چرخشی است که با حرف C نمایش داده می‌شود. در حل دستگاه تاپلیتز $Ax = b$ به روش PCG، پیش شرط استرانگ از روی ماتریس تاپلیتز A بصورت زیر ساخته می‌شود:

۱) سطر اول ماتریس C از روی سطر اول ماتریس A و با استفاده از رابطه $c_k = a_k$ ساخته می‌شود (Chan 1988, 1989، Strang 1986).

۲) بعد از تشکیل سطر اول ماتریس C، بقیه سطرها با استفاده از انتقال به راست چرخشی به اندازهٔ یک مکان از سطرهای قبلی، بدست می‌آید.

ماتریس C ای که به این روش تولید می‌شود به جز در درایه‌های گوشه‌ای، بسیار نزدیک به فرم ماتریس A است. با بکارگیری روش PCG استرانگ، مقادیر ویژه $C^{-1}A$ در اطراف «یک» انباشته می‌گردد که باعث کاهش عدد حالت می‌شود (Chan *et al* 1987).

مثال۲: در حالت $n = 4$ ماتریس A، تاپلیتز و متقارن، ماتریس C به روش استرانگ بصورت زیر ساخته می‌شود:

$$A = \begin{bmatrix} a_0 & a_1 & a_2 & a_3 \\ a_1 & a_0 & a_1 & a_2 \\ a_2 & a_1 & a_0 & a_1 \\ a_3 & a_2 & a_1 & a_0 \end{bmatrix}$$

$$C=\begin{bmatrix} c_0 & c_1 & c_2 & c_1 \\ c_2 & c_0 & c_1 & c_2 \\ c_2 & c_1 & c_0 & c_1 \\ c_1 & c_2 & c_1 & c_0 \end{bmatrix} \xrightarrow{c_k = a_k} \begin{bmatrix} a_0 & a_1 & a_2 & a_1 \\ a_1 & a_0 & a_1 & a_2 \\ a_2 & a_1 & a_0 & a_1 \\ a_1 & a_2 & a_1 & a_0 \end{bmatrix}$$

۲- ۲- معرفی پیش شرط

روش چن در سال ۱۹۸۹ برای حل دستگاه تاپلیتز $Ax=b$ با استفاده از یک پیش شرط بهینه توسط مینیم روش $\min_c \|C-A\|_F$ ارائه شده است که آن را روش PCG چن می‌نامند. پیش شرط در روش چن نیز همانند روش استرانگ یک ماتریس چرخشی است. روش چن در آزمایشات عددی بهتر از روش استرانگ در کاهش عدد حالت ماتریس $C^{-1}A$ عمل می‌کند. قضیه زیر چگونگی ساخت پیش شرط چن را بیان می‌کند (Chan 1988, 1989، Strang 1986، Anisimov 1999).

قضیه۲: فرض کنید A یک ماتریس تاپلیتز و C یک ماتریس چرخشی بصورت زیر باشد:

چرخشی،
$$C=\begin{bmatrix} c_0 & c_1 & \cdots & c_{n-1} \\ c_{n-1} & c_0 & \cdots & c_{n-2} \\ \vdots & & & \vdots \\ c_1 & \cdots & c_{n-1} & c_0 \end{bmatrix}$$

تاپلیتز
$$A=\begin{bmatrix} a_0 & a_1 & \cdots & a_{n-1} \\ a_{-1} & & & a_{n-2} \\ \vdots & & & \vdots \\ a_{-(n-1)} & \cdots & a_{-1} & a_0 \end{bmatrix}$$

در این صورت درایه‌های ماتریس C توسط رابطه زیر داده می‌شود:

$$\min_c \|C-A\|_F \Rightarrow$$
$$c_i=\frac{ia_{-(n-i)}+(n-i)a_i}{n}, i=-(n-1),...,0,...(n-1) \qquad (۲)$$

توجه: به سادگی مشاهده می‌شود که C متقارن است اگر و فقط اگر A متقارن باشد.علاوه بر این $C=A$ است اگر و فقط اگر $a_i=a_{-(n-i)}, i=1.2,...,n-1$. یعنی $C=A$ است اگر A خودش ماتریس چرخشی باشد.

مثال۳: اگر $A_{4\times4}$ ماتریس تاپلیتز و متقارن باشد، ماتریس C به روش چن بصورت زیر داده می‌شود:

از رابطه چن داریم:

$$C=\begin{bmatrix} c_0 & c_1 & c_2 & c_3 \\ c_3 & c_0 & c_1 & c_2 \\ c_2 & c_3 & c_0 & c_1 \\ c_1 & c_2 & c_3 & c_0 \end{bmatrix}=\begin{bmatrix} a_0 & a & a_2 & a \\ a & a_0 & a & a_2 \\ a_2 & a & a_0 & a \\ a & a_2 & a & a_0 \end{bmatrix}$$

$$i=0\Rightarrow c_0=a_0$$
$$i=1\Rightarrow c_1=\frac{a_{-3}+3a_1}{4}=\frac{3a_1+a3}{4}=a$$
$$i=2\Rightarrow c_2=\frac{2a_{-2}+2a_2}{4}=\frac{4a_2}{4}=a_2$$
$$i=3\Rightarrow c_3=\frac{3a_{-1}+a3}{4}=\frac{3a_1+a3}{4}=a$$

تذکر: اگر A تاپلیتز و متقارن باشد رابطه چن را می‌توان بصورت ساده‌تر زیر نوشت:

$$c_i=\frac{ia_{n-1}+(n-i)a_i}{n}, i=0,1,...,n-1 \qquad (۳)$$

۳- حل دستگاه تاپلیتز به روش PCG استرانگ و چن

دستگاه تاپلیتز (۱) را در نظر بگیرید. اگر برای حل این دستگاه از روش PCG با پیش شرطهای استرانگ یا چن استفاده کنیم و پیش شرط را ماتریس C از نوع چرخشی بنامیم آنگاه مراحل زیر را باید انجام دهیم.

۳- ۱- آماده سازی ماتریس پیش شرط

در این مرحله با یکی از روشهای استرانگ یا چن باید ماتریس C را تولید کنیم. با توجه به اینکه ماتریس C شامل n^2 درایه است که باید مقداردهی شوند این مرحله حداکثر در زمان $O(n)$ انجام داد.

۳-۲- حل دستگاه $Cz=r$

با توجه به الگوریتم PCG مشاهده می‌شود که در هر تکرار باید دستگاهی به فرم $Cz=2$ حل گردد. که در آن C ماتریس چرخشی است. و به عنوان پیش شرط استفاده می‌شود. در این جا روش حل این دستگاه را توضیح می‌دهیم. می‌دانیم که اگر $C=F^*\wedge F$، $\wedge=diag(\lambda_0,\lambda_1,...\lambda_{n-1})$ ها مقادیر ویژه C باشند آنگاه $C=Circ(c_0,c_1,...,c_{n-1})$. همچنین از نتیجه همین قضیه میدانیم اگر باشد آنگاه:

$$\lambda_i=\sum_{k=0}^{n-1}c_k w^{kj}, i=0,1,...,n-1 \qquad (۴).$$

از طرفی داریم:

$$C=F^*\wedge F\Rightarrow C^{-1}=(F^*\wedge F)^{-1}=F^{-1}\wedge^{-1}(F^*)^{-1}$$
$$z=C^{-1_r}r,$$

میدانیم:

$$(F^*)^{-1} = F, \quad \wedge^{-1} = diag(1/\lambda_0, 1/\lambda_1 ... 1/\lambda_{n-1})$$

بنابراین می‌توان نوشت :

$$z = (F^* \wedge^{-1} F)r \qquad (۵)$$

رابطه اخیر نشان میدهد که دستگاه $Cz = r$ را می‌توان سه عمل ضرب ماتریس- بردار انجام داد. که در آن ماتریس‌های F^*, F معلوم می‌باشند. اما ماتریس قطری $\wedge^{-1}$ توسط $\lambda_{n-1},...,\lambda_1,\lambda_0$ مشخص می‌گردد. λ_i، مقادیر ویژه ماتریس C هستند که از رابطه (۴) بدست میایند. اگر این رابطه را به فرم ماتریسی در آوریم آنگاه:

$$\begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & w & w^2 & \cdots & w^{n-1} \\ 1 & w^2 & w^4 & & w^{2(n-1)} \\ \vdots & \vdots & & & \vdots \\ 1 & w^{n-1} & w^{2(n-1)} & & w^{(n-1)(n-1)} \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ \vdots \\ c_{n-1} \end{bmatrix} = \begin{bmatrix} \lambda_0 \\ \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_{n-1} \end{bmatrix}$$

بنابراین با توجه به تعریف ماتریس F^* می‌توان نوشت:

$$\begin{bmatrix} \lambda_0 \\ \lambda_1 \\ \vdots \\ \lambda_{n-1} \end{bmatrix} = \sqrt{n} F^* \begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_{n-1} \end{bmatrix}$$

یعنی مقادیر ویژه ماتریس C رامی‌توان با ضرب ماتریس F^* در بردار مربوط به سطر اول ماتریس C بدست آورد که از آنجا ماتریس $\wedge^{-1}$ نیز قابل محاسبه است. در واقع دستگاه $Cz = r$ با چهار عمل ضرب ماتریس – بردار قابل حل است بنابراین پیچیدگی زمانی الگوریتم PCG با فرض اینکه تعداد تکرارهای الگوریتم مقدار ثابتی باشد بصورت زیر در می‌آید: $T_{PCG}(n) = o(k, 4n^2) = o(n^2)$.

۴- کاهش تعداد محاسبات با استفاده از تبدیل سریع فوریه

چنانچه گفته شد در هر تکرار از الگوریتم PCG برای حل دستگاه تاپلیتز $Ax = b$ باید دستگاهی به فرم $Cz = r$ حل گردد. که در آن C ماتریس چرخشی حاصل از استرانگ یا چن می‌باشد. در این قسمت نشان میدهیم که چگونه می‌توان این دستگاه را با استفاده از تبدیل فوریه (FFT: Fast Fourier Transform) با تعداد محاسبات کمتری حل کرد. برای این منظور ابتدا بطور مختصر روش FFT را معرفی می‌کنیم.

۴- ۱- معرفی تبدیل سریع فوریه و الگوریتم‌های آن

اگر $\{a_a, a_1, ...a_{n-1}\}$ یک دنباله از اعداد باشند تبدیل DFT^* آن دنباله $\{b_0, b_1, ..., b_{n-1}\}$ است که بصورت زیر تعریف شده است.

$$b_j = \sum_{k=0}^{n-1} a_k . w^{kj},\ j = 1,2,...,n-1$$

که در آن: $i = \sqrt{-1}, w = e^{\frac{2i\Pi}{n}}$، اگر فرض کنیم $n = 2^s$ آنگاه تبدیل DFT را می‌توان بصورت زیر نوشت:

$$b_j = \sum_{m=0}^{2^{s-1}-1} a_{2m} . w^{2mj} + \sum_{m=0}^{2^{s-1}-1-1} a_{2m+1} w^{(2m+1)j}$$

$$b_j = \sum_{m=0}^{2^{s-1}-1} a_{2m} e\wedge(2\pi jim / 2^{s-1}) + w^j . \sum_{m=0}^{2^{s-1}-1} a_{2m+1} e^{2\pi ijm / 2^{s-1}}$$

که در آن b_j را بصورت مجموع دو DFT نوشته‌ایم و آن را تبدیل سریع فوریه (FFT) می‌نامیم.

روال بازگشتی R-FFt برای محاسبه تبدیل سریع فوریه، دنبالهٔ $A = \{a_0, a_1, ..., a_{n-1}\}$ به عنوان ورودی پذیرفته و دنباله $B = \{b_0, b_1, ..., b_{n-1}\}$ را نتیجه می‌دهد:

$\Pr ocedure\ R - FFT(A, B)$
$if\ n = 1 then\ b_0 \leftarrow a_0$
$else$
$1)\ R - FFT(\{a_0, a_2, ...a_{n-2}\}, \{u_0, u_1, ..., u_{n/2-1}\})$
$2)\ R - FFT(\{a_1, a_3, ...a_{n-1}\}, \{v_0, v_1, ...v_{n/2-1}\})$
$3)\ z \leftarrow 1$
$4)\ for\ j = 0\ to\ n-1\ do$
$4.1)\ b_j \leftarrow u_{j \bmod(n/2)}$
$4.2)\ z \leftarrow z \times w$
$end\ for.$
$end\ if$

برای ارزیابی پیچیدگی زمانی الگوریتم می‌توان نوشت:

$$T(n) = 2T(n/2) + O(n) \Rightarrow$$
$$T(n) = O(n \log n)$$

این الگوریتم در سال ۱۹۶۵ توسط کولی و تاکی ارائه شد که به الگوریتم $Radix - 2$ نیز معروف است. با تبدیل الگوریتم بازگشتی $R - FFT$ به حالت غیر بازگشتی الگوریتم $ITERATIVE - FFT$ به دست می‌آید که در آن X دنباله ورودی و Y دنباله خروجی در نظر گرفته شده است.

$\Pr ocedure\, ITERATIVE-FFT(X,Y,n)$
$Step1:1)\ r \leftarrow \log n$
$1.2)\ for\, i = 0\, to\, n-1\, do$
$endfor.$
$Step\,2: for\ \because m = 0 to : r-1 : do$
$2.1) for\ \because i = 0 to : n-1 : do$
$S[i] \leftarrow R[i]$
$endfor.$
$2.2)\ for\, i = 0\, to\, n-1\, do$
$\{Let\,(b_{r-1}...b_1b_0)\, be\, the\, binary\, representation\, of\ i\}$
$i)\ j \leftarrow (b_{r-1}...b_{m+1}0b_{m-1}...b_0)$
$ii)\ k \leftarrow (b_{r-1}...b_{m+1}1b_{m-1}...b_0)$
$endfor.$
$endfor.$
$Step3: for\, endfor\ \ i = 0\, to\ n-1\ do$
$Y[i] \leftarrow R[i]$
$endfor.$

این الگوریتم دارای log n تکرار در حلقه خارجی از گام ۲ است. مقدار m به عنوان اندیس این حلقه نشان دهنده سطح $\log n - m$ از الگوریتم بازگشتی است. بنابراین الگوریتم مورد نظر از عمیق‌ترین سطح در الگوریتم بازگشتی شروع به کار میکند و مانند هر سطح در الگوریتم بازگشتی در هر تکرار از الگوریتم تعداد n عمل ضرب و n عمل جمع انجام می‌دهد. الگوریتم دارای دو حلقه است در گام ۲ است. حلقه خارجی برای ورودی به طول n دارای $\log n$ تکرار است. حلقه داخلی در هر تکرار از حلقه خارجی انجام می‌شود که خود شامل n تکرار است که هر کدام در زمان ثابتی انجام می‌گیرند. بنابراین زمان الگوریتم برابر $O(n\log n)$است. در هر تکرار از حلقه خارجی دنباله R توسط عناصر ذخیره شده در دنباله S که در تکرار قبلی الگوریتم تولید شده‌اند بهنگام می‌شود. در اولین تکرار دنباله R توسط دنباله X مقدار دهی اولیه می‌شود. در پایان الگوریتم تبدیل FFT دنباله X در دنباله Y تولید می‌شود. اندیس j,k از اندیس i به صورتی که گفته می‌شود به دست می‌آید. فرض کنید $n = 2^r$ و $0 \le m \le r$ دارای نمایش دو دوبی شامل r بیت باشد. اگر $(b_{r-1}...b_1b_0)$ نمایش دو دوبی i باشد در تکرار mام از حلقه خارجی $0 \le m \le r$ اندیس j توسط صفر کردن m امین بیت با ارزش از i بیت b_m می‌تواند صفر یا یک باشد. بنابراین هر بار یکی از دو اندیس j,k با i برابر خواهند بود.

۴-۲- کاهش محاسبات در حل دستگاه تاپلیتز با استفاده از FFT

قضیه ۳: اگر $C_{n\times n}$ ماتریس چرخشی نامنفرد باشد آنگاه دستگاه $Cz = r$ در زمان $O(n\log n)$ قابل حل است.

اثبات: با توجه به قضیه قطری سازی ماتریسهای چرخشی دستگاه $Ca = r$ را می‌توان بصورت زیر حل کرد:

$$\left.\begin{array}{l} z = C^{-1} \\ C = F^* \wedge F \\ \wedge = diag(\lambda_0, \lambda_1, ..., \lambda_{n-1}) \end{array}\right\} \Rightarrow z = (F^* \wedge^{-1} F)r$$

که در آن ماتریسهای F^*, F به ترتیب ماتریس فوریه و ماتریس فوریه ترانسپوزه مزدوج می‌باشند. همچنین λ_i مقادیر ویژه ماتریس C می‌باشند. عمل ضرب ماتریس فوریه یا ماتریس فوریه ترانسپوزه مزدوج در یک بردار معادل با عمل FFT روی بردار مفروض می‌باشد که بجای $O(n^2)$ عملیات میتواند در $O(n\log n)$ انجام گیرد. از طرف دیگر مقادیر ویژه ماتریس C از رابطهٔ (4) بدست می‌آیند که دقیقاً معادل با تعریف DFT میباشد به عبارت دیگر دنباله $\{\lambda_0, \lambda_1, ..., \lambda_{n-1}\}$ را می‌توان از تبدیل سریع فوریه دنباله $\{c_0, c_1, ..., c_{n-1}\}$ در $O(n\log n)$ بدست آورد. بنابراین دستگاه $Cz = r$ در زمان $O(n\log n)$ قابل حل است.

نتیجه: دستگاه $Cz = r$ به صورت $z = (F^* \wedge^{-1} F)$ با استفاده از دو عمل FFT جهت ضرب ماتریسهای F^*F در بردار مفروض و نیز یک عمل FFT جهت محاسبه مقادیر ویژه ماتریس C و در نتیجه محاسبه ماتریس C و در نتیجه محاسبه ماتریس $\wedge^{-1}$، در زمان $O(n\log n)$قابل حل است. توجه کنید که ضرب ماتریس $\wedge^{-1}$ در یک بردار با توجه به قطری بودن آن می‌تواند در $O(n)$ انجام گیرد.

۵- بهینه سازی الگوریتم PCG در حل دستگاه تاپلیتز

در اینجا می‌خواهیم نشان دهیم ضرب ماتریس تاپلیتز در بردار را با استفاده از FFT می‌توان در زمان $O(n\log n)$ به جای زمان $O(n^2)$ انجام داد که به این طریق تعداد محاسبات در هر تکرار از الگوریتم PCG به طور قابل ملاحظه‌ای کاهش خواهد یافت. برای این منظور ابتدا لم زیر را ثابت می‌کنیم (Bunch 1985، Chan *et al.* 1987، Gupta *et al.* 1992).

لم: اگر $T_{n\times n}$ ماتریس تاپلیتز باشد آنگاه ماتریس چرخشی بلوکی $C_{2\times 2}$ وجود دارد بطوریکه $T_{n\times n}$ در جایگاه سطر اول و ستون اول آن قرار دارد.

اثبات: فرض کنید $T_{n\times n}$ ماتریس تاپلیتز باشد. از تعریف ماتریس تاپلیتز نتیجه می‌شود که هر ماتریس تاپلیتز بطور یکتا توسط عناصر سطر اول و ستون اول آن قابل تعریف است. اگر $T_{n\times n}$ را به صورت زیر

در نظر بگیریم.

$$T_{n\times n}=\begin{bmatrix} x_1 & x_2 & \dots & x_n \\ x_{2n-1} & x_1 & \dots & x_{n-1} \\ x_{2n-2} & & \dots & \\ \vdots & & & x_2 \\ x_{n+1} & \dots & x_{2n-1} & x_1 \end{bmatrix}$$

نشان می‌دهیم ماتریس چرخشی C که بصورت زیر تعریف می‌شود همان ماتریس مورد نظر است.

$$C=Circ(x_1,x_2,...,x_nx_{n+1},x_{n+2},...,x_{2n-1})$$

واضح است که ماتریس C به ابعاد $(2n-1)\times(2n-1)$ خواهد بود. اگر ماتریس C را بصورت یک ماتریس بلاکی 2×2 بصورت زیر در نظر بگیریم بطوریکه در آن ماتریسهای C_{11},C_{22} به ابعاد $n\times n$ و ماتریسهای C_{21},C_{12} به ابعاد $(n-1)\times(n-1)$ باشند باید نشان دهیم: $C_{11}=T_{n\times n}$

$$C_{2\times 2}=\begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix}$$

می‌دانیم هر ماتریس چرخشی یک ماتریس تاپلیتز است. بنابراین ماتریس چرخشی C تاپلیتز است در نتیجه هر بلاک از آن تاپلیتز خواهد بود. به عبارت دیگر ماتریس C_{11} تاپلیتز و از مرتبه $n\times n$ است. اکنون برای اثبات تساوی دو ماتریس تاپلیتز $C_{11},T_{n\times n}$ کافیست ثابت کنیم عناصر سطر اول و ستون اول آنها با هم برابر است. با توجه به تعریف C داریم:

$$C=Circ(x_1,x_2,...,x_nx_{n+1},x_{n+2},x_{2n-1})$$

یعنی سطر اول ماتریس C_{11} برابر $(x_1,x_2,...,x_n)$ که برابر با سطر اول ماتریس $T_{n\times n}$ است. از طرفی در هر ماتریس چرخشی هر سطر از چرخش سطر قبلی به اندازه یک مکان به راست پدید می‌آید در نتیجه با توجه به اینکه $n-1$ عنصر آخر از سطر اول C به صورت $(x_{n+1},x_{n+2},...,x_{2n-1})$ در نظر گرفته شده‌اند عناصر $(x_{n-1},x_{n+2},...,x_{2n-1})$ به ترتیب در موقعیت اول از سرهای $(n,n-1,...,2)$ قرار خواهند گرفت. یعنی ستون اول ماتریس C_{11} برابر $(x_1x_{2n-1},...,x_{n+2},x_{n+1})^T$ است که برابر با سطر اول ماتریس $T_{n\times n}$ است. در نتیجه خواهیم داشت $C_{11}=T_{n\times n}$ و اثبات به پایان می‌رسد.

قضیه ۴: ضرب ماتریس تاپلیتز $T_{n\times n}$ در یک بردار میتواند در زمان $O(n\log n)$ انجام شود.

اثبات: فرض کنید $T_{n\times n}$ ماتریس تاپلیتز و x بردار دلخواه $n\times 1$ باشد. و هدف محاسبه $y=Tx$ باشد. با توجه به لم قبل میتوان گفت ماتریس چرخشی بلوکی $C_{2\times 2}$ بصورت زیر وجود دارد.

$$C_{2\times 2}=\begin{bmatrix} T_{n\times n} & C_{12} \\ C_{21} & C_{22} \end{bmatrix}$$

با فرض $\overline{0}=\overbrace{[00...0]}^{n-1}{}^T$ می‌توان نوشت:

$$\begin{bmatrix} T_{n\times n} & C_{12} \\ C_{21} & C_{22} \end{bmatrix}\begin{bmatrix} x \\ \overline{0} \end{bmatrix}=\begin{bmatrix} Tx \\ C_{21}x \end{bmatrix}$$

بردار حاصل از n عنصر اول بردار سمت راست در رابطه بالا همان جواب مورد نظر یعنی y است. بنابراین برای محاسبه $y=Tx$ کافیست ماتریس چرخشی C از مرتبه $(2n-1)\times(2n-1)$ را در بردار $\left[x\overline{0}\right]^T$ ضرب نمائیم که این عمل را میتوان از طریق FFT در زمان $(2n-1)\log(2n-1)$ انجام داد و اثبات قضیه به پایان می‌رسد.

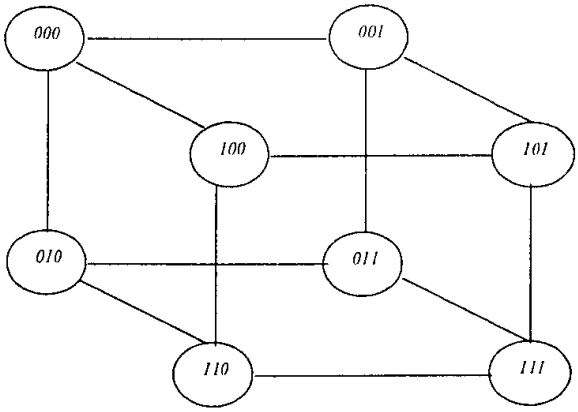
نتیجه: با توجه به اینکه الگوریتم PCG شامل دو عمل اساسی حل دستگاه $Cz=r$ و ضرب ماتریس تاپلیتز در بردار است و هر دو عمل را می‌توان با استفاده از FFT در زمان $O(n\log n)$ انجام داد بنابراین بهترین الگوریتم ترتیبی برای حل دستگاه تاپلیتز $Ax=b$ به روش PCG دارای زمان $O(n\log n)$ است.

۶- الگوریتم‌های موازی FFT

در اینجا روش تعویض دو دویی را برای موازی FFT ارائه می‌کنیم که میتنی بر الگوریتم $ITERATIVE-FFT$ می‌باشد (Kumar *et a.* 1994، Qinn 1990، Strang 1986). در این روش تعویض داده‌ها فقط بین زوج پردازنده‌هایی که در نمایش دودویی تنها در یک بیت با هم اختلاف دارند انجام می‌گیرد. این روش را برای ساختارهای مش و فوق مکعبی در حالتی که هر عنصر ورودی در یک پردازنده در یک پردازنده قرار گیرد و نیز در حالتی چند عنصر ورودی در یک پردازنده قرار گیرد مورد مطالعه قرار می‌دهیم.

۶- ۱- شبکه فوق مکعبی (*Hypercube*)

فرض کنید $P=2^q,q\geq 1,q\in N$ بوده و تعداد P پردازنده، در دسترس باشد. یک شبکه فوق مکعبی با بعد q با ارتباط دادن هر پردازنده با q پردازندهٔ مجاور که هر کدام از لحاظ نمایش دودویی تنها در یک بیت با آن اختلاف دارند به دست می‌آید. توجه داریم که هر پردازنده را می‌توان با نمایش دودویی آن که شامل q بیت است مشخص نمود. چنین شبکه‌ای را می‌توان یک گراف q منتظم نیز تلقی کرد. شکل ۱ این ساختار را برای $p=2^3$ نشان می‌دهد (Anisimov 1999، Akl 1989، Bitmead & Anderson 1980).



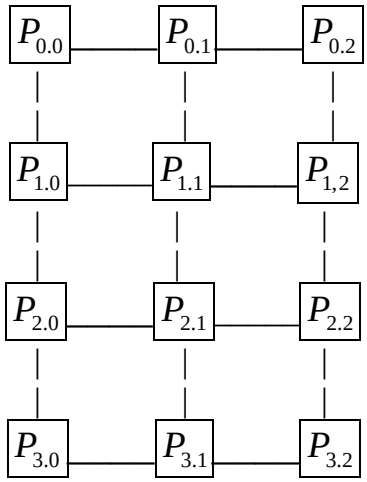
شکل ۱: شبکه فوق مکعبی، P=2J.

الف- هر عنصر در یک پردازنده

در نظر بگیرید در یک شبکه فوق معکبی n پردازنده موجود باشد بطوریکه در هر پردازنده یک عنصر از داده ورودی قرار گرفته باشد. به عبارت دیگر اگر $X[i](i=1,2,...n)$ داده‌های ورودی باشند آنگاه داده $X[i]$ در پردازندهٔ P_i فرار داشته باشد. اکنون الگوریتم $ITERATIVE-FFT$ یا ۲-$Radix$ را در نظر بگیرید. در هر یک از $\log n$ تکرار از حلقه خارجی پردازنده P_i مقدار $R[i]$ را بهنگام می‌کند. برای انجام این عمل P_i به داده‌ای از دنباله S از پردازنده‌ای که در نمایش دودویی تنها در یک بیت با آن اختلاف دارد نیاز دارد. بنابراین محاسبه موازی FFT بطور طبیعی منطبق با ماهیت شبکه فوق مکعبی است. چرا که در این ساختار هر دو پردازنده‌ای که تنها در یک بیت با هم اختلاف دارند بطور مستقیم به هم متصل‌اند. در اولین تکرار از الگوریتم تمام زوج پردازنده‌هایی که در با ارزش‌ترین بیت با هم اختلاف دارند با یکدیگر ارتباط برقرار می‌کنند. بطور مشابه در تکرار دوم الگوریتم تمام زوج پردازنده‌هایی که در دومین بیت با ارزش با هم اختلاف دارند با یکدیگر ارتباط برقرار می‌کنند و به همین ترتیب این روند در تکرارهای بعدی ادامه می‌یابد. در هر تکرار از log n تکرار هر پردازنده یک عمل جمع و یک عمل ضرب و نیز یک عمل ارسال داده به پردازنده‌ای که به طور مستقیم به آن متصل است انجام می‌دهد. بنابراین هر تکرار از الگوریتم زمان ثابتی طول می‌کشد. یعنی زمان الگوریتم موازی FFT در شبکه فوق مکعبی برابر $O(\log n)$ خواهد بود. بنابراین هزینه الگوریتم برابر $O(n\log n)$ خواهد بود. یعنی الگوریتم بهینه است.

ب- چند عنصر در یک پردازنده

فرض کنید بخواهیم تبدیل FFT برای n داده ورودی را بر روی یک شبکه فوق مکعبی با P پردازنده که در آن $P\langle n$ فرض می‌شود به دست آوریم. همچنین فرض کنید P,n هر دو توانی از دو باشند بطوریکه $P=2^d, n=2^r$ دنباله ورودی را به P قسمت به اندازه‌های n/P تقسیم می‌کنیم و هر قسمت را در یک پردازنده قرار می‌دهیم. خاصیت جالب در این روش این است که اگر $b_{r-1}...b_1b_0$ نمایش دودویی i باشد آنگاه $R[i], S[i]$ در پردازنده‌ای قرار دارند که d بیت با ارزش دودویی آن به صورت $b_{d-1}..b_1b_0$ است. این خاصیت نقش مهمی در تعیین تعداد برقراری ارتباط بین پردازنده‌ها در طول اجرای الگوریتم دارد. شکل ۲ این مطلب را برای $P=2^2, n=2^4$ نشان می‌دهد.



شکل ۲: شبکه مش با ابعاد 3×4.

همان طور که می‌دانیم (بنابر اشکال موجود در این Akl 1989) عناصری که اندیس آنها در $d(=2)$ بیت با ارزش با یکدیگر اختلاف دارند در پردازنده‌های جداگانه‌ای قرار دارند. بنابراین تمام عناصری که در d بیت با ارزش یکسانند در یک پردازنده قرار دارند. بیاد دارید که تبدیل FFT برای ورودی به طول n شامل $r=\log n$ تکرار در حلقه خارجی است. در تکرار m ام از حلقه خارجی عناصری که اندیس آنها در m بیت با ارزش با هم اختلاف دارند با هم ترکیب میشوند. در نتیجه عناصری که در d تکرار اول الگوریتم با هم ترکیب می‌شوند در پردازنده‌های یکسانی قرار دارند و زوج عناصری که در جریان $(r-d)$ تکرار باقیمانده با هم ترکیب می‌شوند در پردازنده‌های یکسانی قرار دارند بنابراین در این الگوریتم ارتباطات بین پردازنده‌ای فقط در جریان $d=\log P$ تکرار از $\log n$ تکرار انجام می‌گیرد و در جریان $(r-d)$ تکرار دیگر هیچ ارتباط بین پردازنده‌ای انجام نمی‌گیرد به عبارت دیگر در i امین تکرار را از اولین d تکرار تمام عناصری که یک پردازنده به آنها نیاز دارد از پردازنده‌ای می‌باشند که در نمایش دودویی در i بیت با ارزش با این پردازنده اختلاف دارند. در هر عمل ارتباطی تعداد n/P عناصر بین دو پردازنده منتقل می‌شوند. از طرفی هر عمل ارتباطی فقط بین دو پردازنده‌ای که بطور مستقیم به هم متصل‌اند انجام می‌شود. بنابراین زمان عمل ارتباطی

فقط بین دو پردازنده‌ای که بطور مستقیم به هم متصل‌اند انجام می‌شود. بنابراین زمان عمل ارتباطی در کل برابر $t_s \log P + t_w (n/P)\log P$ خواهد بود. از طرفی هر پردازنده در طول $\log P$ تکرار تعداد n/P از عناصر R را بهنگام می‌کند. اگر عملیات جمع و ضرب مختلط دارای زمان ثابت t_c باشد زمان اجرای الگوریتم موازی FFT در شبکه فوق مکعبی با n ورودی و P پردازنده بصورت زیر در می‌آید.

$$T_p = t_c \frac{n}{P}\log p + t_s \log P = O(\frac{n}{P}\log n)$$

بنابراین هزینه الگوریتم برابر است با:

$$Cost = P.T_p =$$

$$t_c n \log P + t_s P \log P + t_w n \log P = O(n \log n)$$

یعنی الگوریتم بهینه است. سرعت و کارآیی الگوریتم میتواند بصورت زیر بیان شود.

$$S = \frac{t_c n \log n}{T_P} =$$

$$\frac{Pn\log n}{n\log n + (t_s/t_c)P\log P + (t_w/t_c)n\log P}$$

$$E =$$

$$\frac{1}{(1+t_S P\log P)/(t_C n\log n) + (t_W \log P)/(t_C \log n)}$$

۷- ۲- شبکه مش (*Mesh*)

یک شبکه مش شامل $P = r \times c$ پردازنده است که در آن پردازنده‌ها در r سطر و c ستون سازماندهی شده‌اند. $P_{i,j}$ پردازنده‌ای است که در سطر iام با پردازنده‌های $P_{i-1,j}$ و $P_{i,j+1}$ و $P_{i,j-1}$ و $P_{i+1.j}$ در صورت وجود متصل است. شکل ۲ یک شبکه دو بعدی با ابعاد 3×4 را نشان می‌دهد (Miranker 1971، Saad 1988).

فرض کنید برای محاسبه FFT یک دنباله n در ساختار مش از P پردازنده که در $\sqrt{P}$ سطر و $\sqrt{P}$ ستون سازماندهی شده‌اند استفاده شود. بطوریکه $\sqrt{P}$ توانی از دو باشد. همچنین فرض کنید $n = 2^r$ و $P = 2^d$ باشد پردازنده‌ها بصورت ترتیب سطری شماره‌گذاری می‌شوند. توزیع داده‌ها در پردازنده‌ها به همان شکلی که در شبکه فوق مکعبی بیان شد انجام می‌گیرد. در واقع عنصری با اندیس دودویی $b_{r-1}...b_1b_0$ در پردازنده‌ای با نمایش دودویی $b_{d-1}...b_1b_0$ قرار می‌گیرد. همانند آنچه در شبکه فوق مکعبی داشتیم ارتباطات بین پردازنده‌ای فقط در طول اولین logP تکرار از الگوریتم بین پردازنده‌هایی که اندیس آنها در یک بیت با هم اختلاف دارند انجام می‌گیرد. از طرفی برخلاف شبکه فوق مکعبی این پردازنده‌ها در ساختار مش بطور مستقیم به هم متصل نیستند. به این ترتیب تبادل پیام بین دو پردازنده از طریق بیش از یک اتصال انجام خواهد گرفت. بطور کلی‌تر میتوان گفت در $\log\sqrt{P}$ گام از $\log P$ گام لازم برای ایجاد ارتباط پردازنده‌های شرکت کننده در ارتباط در یک سطر قرار دارند و در $\log\sqrt{P}$ گام باقیمانده این پردازنده‌ها در یک ستون قرار دارند. فاصله بین این پردازنده‌ها در یک سطر یا یک ستون در هر یک از $\log\sqrt{P}$ گام دو برابر می‌شود در نتیجه این فاصله از یک اتصال به $\sqrt{P/2}$ اتصال رشد می‌یابد. بنابراین مجموع زمان لازم برای انجام ارتباطات سطری درمش به روش SF برابر است با:

$$\sum_{m=0}^{d/2-1}(t_s + t_w(n/P)2^m)$$

مجموع زمان لازم برای ارتباطات ستونی نیز برابر همین مقدار است. اگر زمان لازم برای عمل جمع و ضرب مختلط را با t_c نمایش دهیم با توجه به اینکه درهر تکرار از $\log n$ تکرار، هر پردازنده عملیات لازم را برای n/P عناصر انجام می‌دهد بنابراین زمام اجرای الگوریتم موازی برابر است با:

$$T_P = t_c \frac{n}{P}\log n + 2\sum_{m=0}^{d/2-1}(t_s + t_w \frac{n}{P})2^m$$

$$= t_c \frac{n}{P}\log n + 2(t_s \log\sqrt{P} + t_w \frac{n}{P}(\sqrt{P}-1)$$

$$\approx t_c \frac{n}{P}\log n + t_s \log P + 2t_w \frac{n}{\sqrt{P}} = O(\frac{n}{P}\log n)$$

هزینه الگوریتم برابر است با:

$$Cost = P.T_p =$$

$$t_c n\log n + t_s P\log P + 2t_w n\sqrt{P} = O(n\log n)$$

بنابراین الگوریتم با شرط $P = O(\log^2 n)$ یا $\sqrt{P} = O(\log n)$ بهینه است. سرعت و کارآیی الگوریتم را می‌توان به صورت زیر بیان کرد:

$$E = \frac{1}{1 + (t_s P\log P)/(t_c n\log n) + 2(t_w\sqrt{P})/(t_c \log n)}$$

$$S = \frac{t_c n\log n}{T_p} = \frac{Pn\log n}{n\log n + (t_s/t_c)P\log P + 2(t_w/t_c)n\sqrt{P}}$$

۷- الگوریتم موازی PCG در شبکه‌های مش و فوق مکعبی

اکنون با داشتن الگوریتم‌های موازی FFT در شبکه مش و فوق مکعبی براحتی میتوان الگوریتم موازی PCG را در این ساختار طرح‌ریزی نمود. دستگاه تاپلیتز $Ax = b$ را که در آن $A_{n\times n}$ ماتریس تاپلیتز، حقیقی و متقارن معین مثبت است در نظر بگیرید. این ماتریس با عناصر قطری $a_0, a_1, ..., a_{n-1}$ مشخص می‌شود. فرض کنید

$P = 2^{2S}$ و تعداد P پردازنده شامل $P_0, P_1, ..., P_{P-1}$ در شبکه مش یا فوق مکعبی موجود باشد. همچنین فرض کنید n مضربی از P باشد. اکنون بردارهای $a = [a_0, a_1, ..., a_{n-1}]$، $\bar{a} = [a_0, a_{n-1}, ..., a_2, a_1]$ را که از روی عناصر سطری ماتریس تاپلیتز A ساخته می‌شوند در نظر بگیری.د. هر کدام از این دو بردار را به P قسمت تقسیم کرده و هر قسمت را که شامل $\frac{n}{P}$ عنصر است در یک پردازنده قرار می‌دهیم. با توجه به این مفروضات در ادامه عملیات لازم برای حل دستگاه را توضیح می‌دهیم.

الف- آماده سازی ماتریس پیش شرط

در این جا نحوهٔ آماده سازی پیش شرط چن را که یک ماتریس چرخشی است توضیح می‌دهیم. همانطور که می‌دانیم طبق روش چن درایه‌های ماتریس C توسط فرمول (2) زیر داده می‌شود (A متقارن است).

الگوریتم موازی $FSETUP - PRECOND$ پیش شرط نوع چن را که یک ماتریس چرخشی بصورت $C = Circ(c_0, c_1, c_{n-1})$ است، تولید و آماده سازی می‌کند:

$\Pr ocedure\ F - SETUP - PRECOND(A, C)$
$for\ i = 1\ to\ P-1\ do\ in\ paralled$
$\Pr ocessor\ P_i\ do$
$for\ \ j = 0\ to(\frac{n}{P} - 1)\ do$
$2.1)\ k \leftarrow i \times (\frac{n}{P} - 1) do$
$2.2)\ if\ k = 0\ then\ c_0 \leftarrow a_0$
$else\ c_j \leftarrow \frac{k\bar{a}_i + (n-k)a_j}{n}$
$end\ if$
$end\ for.$
$end\ for.$

توجه کنید که برای محاسبهٔ c_1 به دو دادهٔ a_{n-i} و a_1 نیاز است که هر دوی آنها با توجه به نوع توزیع بردارهای $a, \bar{a}$ در پردازنده مورد نظر موجود می‌باشند. ابتدا در گام 2.1 با استفاده از رابطه $k = i \times \frac{n}{P} j$ اندیس واقعی عنصر مورد نظر پیدا شده و سپس در گام 2.2 با استفاده از فرمول چن مقدار c_j محاسبه میشود. واضح است که الگوریتم در زمان $O(\frac{n}{P})$ انجام می‌گیرد یعنی هزینه آن برابر $O(n)$ بوده و بهینه است. علاوه بر این میزان حافظه مورد استفاده در این روش برابر $O(2n)$ می‌باشد که از لحاظ مصرف حافظه نیز بهینه است. که علت آن ذخیره سازی عناصر سطری ماتریسها به جای کل ماتریس است.

ب- حل دستگاه $Cz = r$

با توجه به رابطه $z = (F^* \wedge^{-1} F)r$ و نیز در نظر گرفتن این مطلب که مقادیر ویژه ماتریس C از طریق تبدیل سریع فوریه محاسبه است. الگوریتم موازی حل دستگاه $Cz = r$ را می‌توان براحتی طرح‌ریزی نمود (Saad 1988، Vagda 1993، Misra et al. 1993، Joubert 1992، Kumar 1982، Heller *et al* 1978، Bitmead & Anderson 1980، Demmel, et al1993).

فرض کنیم بردارهای r, z در پردازنده‌های $P_0, P_1, ..., P_{P-1}$ به صورت چند مولفه در یک پردازنده توزیع شده باشند.

بطوریکه در هر پردازنده تعداد $\frac{n}{P}$ از عناصر دو بردار قرار گرفته باشند. همچنین به همین صورت λ_iها نیز در پردازنده‌ها توزیع شده باشند. الگوریتم دارای مراحل زیر است.

۱- انجام عمل FFT معکوس روی دنباله $\{r_0, r_1, ...r_{n-1}\}$ و ذخیره تبدیل در $\{z_0, z_1, ..., z_{n-1}\}$

۲- با فرض اینکه ماتریس قطری $\wedge^{-1}$ قبلا توسط عمل FFT بر روی دنباله $\{c_0, c_1, ..., c_{n-1}\}$ و ذخیره تبدیل در $\{\lambda_0, \lambda_1, ..., \lambda_{n-1}\}$ و سپس قرار دادن $\lambda_i = 1/\lambda_i$ و $i = 0,1,..., n-1$ و محاسبه شده است. عمل ضرب ماتریس قطری $\wedge^{-1}$ در نمودار Z را بصورت زیر انجام می دهیم. $z_i \leftarrow z_i / \lambda_i$ i=0,…,n-1

۳- انجام عمل FFT روی دنباله $\{z_0, z_1, ..., z_{n-1}\}$ و قرار دادن $i = 0,1,..., n-1,\ z_i \leftarrow z_i / n$ که نتیجه آن جواب دستگاه $Cz = r$ می‌باشد.

همانطور که می‌دانیم ضرب ماتریس F^* در بردار معادل با عمل FFT و ضرب ماتریس F در یک بردار معادل با عمل FFT معکوس میباشد. واضح است که هر دو عمل ماهیتا یک تبدیل فوریه محسوب می‌شوند و تنها تفاوت در توانهای w. ما برای تمایز این دو عمل آنها را با $IFFT, FFT$ نشان می‌دهیم. الگوریتم موازی $FSOLVE$ بصورت زیر نوشته شود:

$\Pr ocedure\ FSOLVE(c, z, r)$
$Step1: IFFT(r, z)$
$Step2: for\ i = 0\ to\ (P-1)\ do\ in\ parallel$
$for\ \ j = 0\ to(\frac{n}{P} - 1)\ do$
$z_j \leftarrow z_j / \lambda_j$
$end\ \ for.$
$end\ \ for.$

$Step3:3.1)\ FFT(z,z)$

3.2) $for\ i=0\ to\ P-1\ do\ in\ paralle$

$for\ j=0 to (\frac{n}{P}-1) do$

$z_j \leftarrow z_j / \lambda_j$

$end\ for.$

$end\ for.$

آنالیز الگوریتم: گامهای 3.2,2 در زمان $O(\frac{n}{P})$ انجام می‌گیرند. در گامهای 3.1,1 الگوریتم موازی FFT اجرا می‌شود که اگر از الگوریتم‌های بهینه تعویض دودویی یا ترانهاده استفاده کنیم زمان آن در حالت چند عنصر در یک پردازنده برابر $O(\frac{n}{P}\log n)$ خواهد بود. بنابراین خواهیم داشت:

$$T_P = O(\frac{n}{P}\log n)$$

$$Cost = P\,T_P = O(n\log n)$$

یعنی الگوریتم بهینه است. سرعت و کارآیی الگوریتم بصورت زیر به دست می‌آید.

$$S = \frac{T_{Seq}}{T_P} = \frac{O(n\log n)}{O(\frac{n}{P} lig\ n)} = O(P)$$

$$E = \frac{S}{P} = O(1)$$

چ- ضرب ماتریس تاپلیتز در بردار

یکی از عملیات اساسی که در الگوریتم PCG جهت حل دستگاه تاپلیتز انجام می‌شود عمل ضرب ماتریس تاپلیتز در بردار مفروض می‌باشد. از قضیه ۴ می‌دانیم که ضرب ماتریس تاپلیتز در یک بردار با استفاده از جاسازی ماتریس تاپلیتز در یک ماتریس چرخشی از طریق سه عمل FFT بصورت موازی در ساختارهای مش و فوق مکعبی انجام داد. اگر ماتریس تاپلیتز A با مقادیر $a_0, a_1, ..., a_{n-1}$ مشخص شود و بردار V با مؤلفه‌های $v_0, v_1, ..., v_{n-1}$ در دست باشد. آنگاه ابتدا بردار V در پردازنده‌های $P_0, P_1, ..., P_{P-1}$ بصورت چند مولفه در یک پردازنده توزیع می‌شود. بطوریکه در هر پردازنده $\frac{n}{P}$ عنصر موجود باشد. از قضیه ۴ ماتریس چرخشی $C' = Circ(a_0, a_1, ..., a_{n-1}, 0, ..., a_2 a_1)$ که یک ماتریس چرخشی با ابعاد $(2n)\times(2n)$ برای این منظور در نظر گرفته می‌شود. عناصر سطر اول این ماتریس بصورت یک بردار در پردازنده‌های $P_0, P_1, ..., P_{P-1}$ بصورت چند مولفه در یک پردازنده توزیع می‌شوند بطوریکه در هر پردازنده $\frac{2n}{P}$ از عناصر باشد. اکنون اگر ماتریس چرخشی C' را بصورت $C'F^*\Delta^{-1}F$ قطری سازی نماییم که در آن $\Delta = \{\mu_o, \mu_1, ..., \mu_{2n-1}\}$ و μ_i ها مقادیر ویژه ماتریس C' می‌باشند آنگاه عمل ضرب ماتریس تاپلیتز در بردار V معادل با عمل ضرب $(F^*\Delta^{-1}F)\begin{bmatrix} V \\ 0 \end{bmatrix}$ خواهد بود که از طریق FFT همانند آنچه برای حل دستگاه $Cz = r$ گفته شد قابل انجام است ($\bar{0}$ یک بردار n تایی با مولفه‌های صفر است). توجه دارید که در بردار $2n$ عنصری حاصل تنها n عنصر اول بردار که در $\frac{P}{2}$ پردازنده‌های ابتدایی قرار گرفته‌اند جواب مسأله است. با توجه به آنچه گفته شد نتیجه می‌شود عمل ضرب ماتریس تاپلیتز در یک بردار قابل انجام صورت موازی بر روی شبکه‌های مش و فوق مکعبی در زمان $O(\frac{2n}{P}\log(2n))$ است.

۸- ارزیابی الگوریتم موازی PCG در شبکه مش و فوق مکعبی

الگوریتم موازی PCG شامل عملیات اساسی زیر است.

T_{seutp} = زمان آماده سازی ماتریس پیش شرط

$T_{solve} = Cz = r$ زمان حل دستگاه

$T_{mult=}$ زمان ضرب ماتریس تاپلیتز در بردار

بنابراین میتوان نوشت:

$$T_p = T_{setup} + T_{solve} + T_{Mult}$$

$$T_p O(\frac{n}{P}) + O(\frac{n}{P}\log n) + O(\frac{2n}{P}\log(2n))$$

$$T_P = O(\frac{n}{P}\log n)$$

هزینه الگوریتم برابر است با:

$$Cost = P\,T_P = O(n.\log n)$$

الگوریتم بهینه است. و داریم:

$$S = \frac{T_{seq}}{T_{P_p}} = \frac{O(n\log n)}{O(\frac{n}{P}\log n)} = O(P)$$

یعنی الگوریتم علاوه بر بهینه بودن دارای سرعتی معادل P برابر الگوریتم ترتیبی PCG است.

۹- نتایج عددی

با پیاده سازی الگوریتم‌های بیان شده در PVM و محاسبه زمان اجرای هر تکرار، سرعت و کارآیی برنامه نتایج به دست آمده مطابق جداول ۱ و ۲ و ۳ می‌باشد. فرمول‌های محاسبه سرعت و کارآیی بصورت زیر میباشد. که در آن T_1, T_P به ترتیب برابر زمان اجرای الگوریتم موازی

با P پردازنده و زمان اجرای الگوریتم با یک پردازنده می‌باشند.

$$(Speedup)\,S = \frac{T_P}{T_1}$$

$$(Effeciency)\,E = \frac{T_1}{P\,T_P}$$

جدول ۱: زمان هر تکرار از الگوریتم Parallel PCG به ملی ثانیه

	تعداد پردازشگرها					
N	1	2	4	8	16	32
256	0.926	0.608	0.414	0.353	0.303	0.399
512	1.721	1.044	0.414	0.463	0.384	0.371
1024	3.608	2.082	0.688	0.747	0.543	0.488
2048	6.011	4.646	1.170	1.143	0.929	0.714
4096	10.649	5.551	2.386	1.592	1.421	0.997

جدول۲: سرعت (Speedup) الگوریتم Parallel PCG

	تعداد پردازشگرها					
N	1	2	4	8	16	32
256		1.374	2.019	2.368	2.757	2.092
512		1.648	2.283	3.718	4.481	4.635
1024		1.646	2.927	4.585	6.304	7.019
2048		1.188	2.314	3.855	5.943	7.732
4096		1.769	3.363	6.167	6.909	9.848

جدول ۳: کارآیی (Efficiency) الگوریتم Parallel PCG

	تعداد پردازشگرها					
N	1	2	4	8	16	32
256		0.687	0.505	0.296	0.172	0.065
512		0.823	0.571	0.467	0.280	0.145
1024		0.823	0.732	0.573	0.394	0.219
2048		0.594	0.578	0.482	0.371	0.242
4096		0.884	0.891	0.771	0.432	0.308

منابع:

Akl S.G. 1989: The design and analysis of parallel algorithms. Prentice Hall Inter. Inc.

Anisimov A.V. 1999: Fibonacci hypercube. *Inter.J. of Comp. Math.* **25:** 221-227.

Berteskas D.P., Tsitsiklis J.N. 1989: Parallel and distributed computations. Prentice Hall.

Bitmead R.P., Anderson B.D. 1980: Asymptotically fast solution of methods for Toeplitz and related systems of equations, Linear. *Algebra and Appl.* **34:** 103-116.

Bunch J.R. 1985: Stability of methods for solving Toeplitz system of equations. *SIAM Sci. Stat. Comput.* **6:**.137-142.

Chan R.H. 1988: An optimal circulant preconditioners for solving Toeplitz system of equations. *SIAM J. Sci. Comput.* **9:** 541-634.

Chan R.H., Strang G. 1987: The asymptotic Toeplitz-circulant eigenvalue problem. *Appl. Math*. **8:** 210-231.

Chan R.H., Strang G. 1989: Toeplitz equation by conjugate gradients with circulant preconditioners. *SIAM J. Sci Comput.* **10:** 104-119.

Davis D. 1979: Circulant Matrices. John Wiley & Sons. New York.

Demmel J.W., Health M.T. 1993: Vander H.A. Vort, Parallel numerical algebra. *Acta Numerical*.**5:** 197-111.

Gallivan K., Plemmons R.J., Sameh A.H. 1989: Parallel algorithms for dense linear algebra a computations. Prentice Hall.

Gupta A., Kumar V., Sameh A. 1992: Performance and scalability of preconditioned conjugate gradients methods on parallel computers, Tech. Rep. TR. 92-6, Dep. of computer Sci. University of Texas at Austin.

Heller D. 1978: A. Survey of parallel algorithms in numerical linear algebra. *SIAM Review*. **20:** 740-777.

Joubert W. 1992: Lanczos methods for the solution of non-symmetric system of linear equations, *SIAM J. Sci. Matrix Anal. Appl.* **13:** 926-943.

Kumar S.P. 1982: Parallel algorithms for solving linear equations on MIMD computers, Ph.D. Thesis. Dept. of Comput. Sci University of Minnesota.

Kumar V., Grama A., Gupta A., Karypis G. 1994: Introduction to parallel computing, The Bengamin Cummings Publishing Company. Inc.

Miranker W.L. 1971: A survey of parallelism in numerical analysis. *SIAM Review*. **13:** 547-524.

Misra M., Nassimi D., Prasanna-Kumar V. K. 1993: Efficient VLSI implementation of iteratives solutions of sparse linear equations. *Parallel Computing.* **19:** 525-544.

Qinn M.J. 1990: Parallel computing. Second Edition. Prentice Hall.

Saad Y. 1988: Krylov subspace methods on super computers. *SIAM J. Sci. Stat. Comput.* **10:** 203-228.

Saad Y. 2002: Iterative methods for sparse linear system, Second Edition, SIAM Publication.

Sameh A.H., Kuck D. J. 1977: Parallel direct linear system. Feilmeier. 25-30.

Strang G. 1986: A proposal for Toeplitz matrix Calculations. *Stud. Appl. Math*. **74:** 171-176.

Vagda S. 1993: Fibonacci & Lucas number and the golden section theory and applications. Prentice Hall.