ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 1989

Seria: TRANSPORT z. 14 Nr kol. 1048

Roman BĄK
Tomasz MATYJA

Instytut Transportu
Politechniki Śląskiej, Katowice

PROCEDURY UKŁADANIA I ROZWIĄZYWANIA DUŻYCH UKŁADÓW RÓWNAŃ UZUPEŁNIAJĄCE SYSTEM PROGRAMÓW KOŁO-PC

Streszczenie. W pracy zaprezentowano pakiet procedur służący do generowania i rozwiązywania dużych układów równań z macierzami symetryczmymi i pasmowymi, dostosowany do możliwości minikomputera typu IBM-PC. Pakiet stanowi uzupełnienie dla programów systemu KOŁO-PC przeznaczonych do analizy stanu naprężenia w zestawach kołowych pojazdów szynowych. Umożliwia rozwiązywanie znacznie większych zadań MES niż dotychczas analizowane programami systemu. Rozmiar macierzy sztywności może kilkakrotnie przekraczać dostępną pamięć RAM.

W pakiecie wykorzystano algorytm Choleskiego-Banachiwicza, który zmodyfikowano ze względu na zastosowanie własnych, maksymalnie efektywnych metod dostępu do elementów tablic.

Autorzy przedstawili również koncepcję budowy modułu "algebry liniowej" przeznaczonego do szerokiego wykorzystania w procesach MES i MRS. Stworzenie takiego modułu okazało się niezbędne i wynika z konieczności rozszerzenia systemu KOŁO w kierunku nowych zagadnień, np. dynamiki zestawów kołowych.

1. SFORMUŁOWANIE PROBLEMU

Podczas adaptacji SYSTEMU KOŁO [1] na minikomputery typu IBM-PC wykonano szereg zmian i ulepszeń, modernizując głównie metody wprowadzania danych i edycji wyników oraz tworząc tzw. przyjazne otoczenie dla użytkownika.

Dalsza modernizacja i rozbudowa systemu wymagała będzie poczynienia zmian w algorytmach obliczeniowych. Jednym z problemów, który wyłonił się w trakcie prac koncepcyjnych nad kolejną, przebudowaną wersją systemu Koło, były ograniczenia wielkości zadań (liczba węzłów) jakie można rozwiązywać za pomocą minikomputera.

Problem ten jest bardziej ogólny i wiąże się z trudnościami jakie napotykamy podczas rozwiązywania dużych układów równań na minikomputerach. Wspomniane trudności wynikają nie tylko z rozmiarów dostępnej pamięci operacyjnej, ale są bezpośrednio związane ze sposobem adresowania mikroprocesorów rodziny 8086. Niektóre kompilatory (np. Turbo Pascal 4.0) nie pozwalają programiście lokować w pamięci obiektów większych niż 64 kB, inne (np. Fortran 4.0, MS-C) umożliwiają wprawdzie wybór modelu pamięci

(huge), przy którym ograniczenia te nie obowiązują, ale związane jest to ze znacznym wzrostem czasu dostępu do elementów tablic (skalowanie adresów).

Odczuwalny jest brak odpowiedniego, dostępnego oprogramowania, które można by łatwo adaptować przy pisaniu własnych programów. W związku z tym opracowano pakiet procedur umożliwiających sprawne generowanie oraz rozwiązywanie dużych układów równań. W pierwszej fazie ograniczono się do przypadku gdy macierz układu jest symetryczna, pasmowa i dodatnio określona. Jednak już w trakcie opracowywania pakietu powstał zamysł budowy całego modułu "algebry liniowej" (rozwiązywanie układów z macierzą niesymetryczną, zagadnienie własne dla macierzy symetrycznych i niesymetrycznych), który mógłby mieć szersze znaczenie w problemach MES i MRS.

Jako wyjściowy algorytm rozwiązujący przyjęto odmianę metody Choleskiego-Banachiewicza prowadzącą do rozkładu macierzy A układu równań na iloczyn LDL^T, gdzie L macierz trójkątna dolna, D - macierz diagonalna [2]; którą zmodyfikowano i przystosowano dla przypadku macierzy pasmowych. Za wybraną metodą przemawiały:

- prostota algorytmu oraz jego dostosowanie do charakteru macierzy MES,
- korzystne cechy, jakie ujawniły się w trakcie transformacji algorytmu dla macierzy pasmowej,
- fakt uzyskiwania rozkładu LDL^T, pozwalający rozwiązywać wiele układów równań różniących się wektorem wyrazów wolnych, w czasie znacznie krótszym niż sam proces rozkładu macierzy (restart w MES).

W pakiecie przewidziano możliwość iteracyjnego poprawiania rozwiązania uzyskanego wspomnianą wyżej metodą. Zrezygnowano z typowych mechanizmów dostępu do elementów tablic, ze względu na ich małą elastyczność i czas. Stworzono własne, maksymalnie szybkie (ze względu na adresowanie) algorytmy dostępu, zapewniające pełną dynamikę procedur.

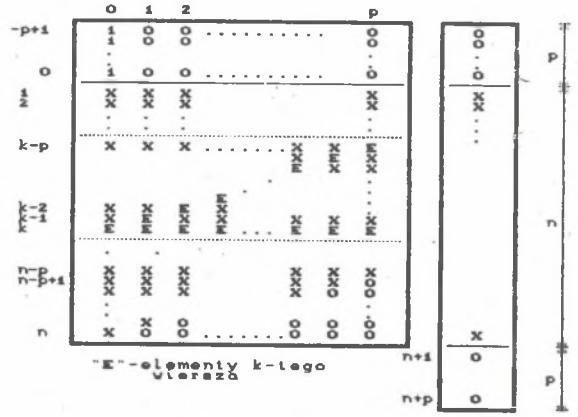
2. BUDOWA MACIERZY I WEKTORÓW

Specyfika algorytmów dla macierzy pasmowych wymaga odpowiedniej budowy macierzy i wektorów:

1° Ze względu na oszczędność miejsca macierz a_{ij} symetryczna i pasmowa zapamiętana jest jako półpasmo górne (pasmo=2p+1), tzn. jako macierz A_{ij} o wymiarach n(p+1), gdzie:

$$a_{ij} = \begin{cases} A_{i,j-i} & i \leqslant j \leqslant i+p \\ a_{ji} = A_{j,i-j} & i-p \leqslant j \leqslant i \\ 0 \quad \text{w pozostałych przypadkach} & \end{cases} \tag{1}$$

2^{o} W celu zapewnienia płynności algorytmów macierz ta musi być powiększona o dodatkowe p wierszy "zerowych" i ostatecznie ma postać pokazaną na rysunku i wymiar (n+p)(p+1). Przy wyraźnym paśmie koszt pamiętania dodatkowych wierszy jest pomijalnie mały. Z podobnych przyczyn również wszystkie wektory muszą mieć dodatkowe p zer na początku i na końcu (rys. 1).



Rys. 1. Budowa macierzy i wektorów układu równań

Fig. 1. Structure of matrice and vectors system of linear equations

3^{o} Postulat pełnej dynamiki sprawia, że faktycznie macierz zapamiętana jest w postaci wektora **A** o wymiarach $[0..(n+p)(p+1)-1]$, co łącznie z 1^{o} i 2^{o} prowadzi do następujących wzorów transformacyjnych:

dla macierzy $$A_{\alpha\beta} \rightarrow \mathbf{A}_{(p+\alpha-1)(p+1)+\beta} \qquad (2)$$

dla macierzy $$X_{\phi} \rightarrow \mathbf{X}_{p+\phi-1} \qquad (3)$$

3. KRÓTKI OPIS ALGORYTMÓW

3.1. Metoda Choleskiego-Banachiewicza

Rozkład $\mathbf{L}\,\mathbf{D}\,\mathbf{L}^{T}$ macierzy **A** uzyskuje się za pomocą następującego algorytmu:

$i=1...n$

$$\mathbf{A}_{(p+i-\alpha-1)+\alpha} = \mathbf{A}_{(p+i-\alpha-1)(p+1)+\alpha} + \\ - \sum_{\beta=p}^{\alpha+1} \mathbf{A}_{(p+i-\beta-1)(p+1)+\beta}\, \mathbf{L}_{(p+i-\beta-1)+\beta-\alpha}; \qquad \alpha=p..1$$

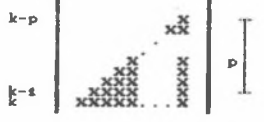
$$\mathbf{A}_{(p+i-1)(p+1)} = \mathbf{A}_{(p+i-1)(p+1)} +$$

$$- \sum_{\beta=p}^{1} \mathbf{A}_{(p+i-\beta-1)(p+1)+\beta} \mathbf{L}_{(p+i-\beta-1)(p+1)+\beta} ; \qquad (4)$$

gdzie

$$\mathbf{L}_{(p+i-\alpha-1)(p+1)+\alpha} = \mathbf{A}_{(p+i-\alpha-1)(p+1)+\alpha} / \mathbf{A}_{(p+i-\alpha-1)(p+1)}$$

Po wykonaniu wszystkich kroków algorytmu (4) macierz wyjściowa **A** ulega zniszczeniu a na jej miejscu znajduje się macierz trójkątna górna $\mathbf{L}^T$ oraz macierz diagonalna **D**. Można zauważyć, że przy rozkładzie k-tego wiersza algorytm odwołuje się tylko do elementów z wiersza k oraz do elemetnów p wierszy leżących powyżej wiersza oznaczonego numerem k (por. rys. 2).



Rys. 2. Elementy macierzy wykorzystywane w k-tym kroku algorytmu

Fig. 2. Elements of matrice used in k-step of algorithm

Powyższa własność oznacza, że w danej chwili w pamięci operacyjnej potrzebna jest możliwość dostępu tylko do części macierzy. Pozwala to rozszerzyć algorytm na przypadek dużych macierzy, nawet takich, które nie mieszczą się w dostępnej pamięci (na stercie) i w całości pamiętane są na dysku. Ponadto możliwe jest ominięcie trudności związanych z ograniczeniami na wielkość obiektów dynamicznych (do 64 kB w Turbo Pascalu) i stworzenie metod szybkiego dostępu (adresowania) wyłącznie do potrzebnej na danym etapie obliczeń części macierzy.

Liczba działań algorytmu wynosi około $np^2/2$, co w porównaniu z algorytmem klasycznym $(n^3/6)$ oznacza, że algorytm zmodyfikowany opłacalny jest gdy $p < n/2$. W praktyce znacznie silniejsze ograniczenia na pasmo nakłada dostępna pamięć RAM (por. pkt 4).

3.2. Metoda rozwiązywania układu równań

Układ równań rozwiązuje się w dwóch etapach:

etap 1 $\mathbf{L}\,\mathbf{Y} = \mathbf{B}$ (z macierzą trójkątną dolną)

$i = 1..n$

$$Y_{(p+i-1)} = B_{(p+i-1)} - \sum_{\alpha=p}^{1} A_{(p+i-\alpha-1)(p+1)+\alpha} \, Y_{(p+i-1-\alpha)} \qquad (5)$$

etap 2 $D L X^T = Y$ (z macierzą górną)

$i = 1..n$

$$X_{(p+i-1)} = Y_{(p+i-1)} / A_{(p+i-1)(p+1)} + \\ - \sum_{\alpha=1}^{p} A_{(p+i-1)(p+1)+\alpha} \, X_{(p+i-1+\alpha)} \qquad (6)$$

Każdy z etapów wymaga około np działań (klasyczny algorytm $(n^2-n)/2$ działań).

3.3. Iteracyjne poprawianie rozwiązania

Korzystamy tu z następującego algorytmu [3], zakładającego istnienie rozwiązania początkowego x_o, które uzyskano metodą ϕ :

$i=0,1,...$

1^o Obliczamy wektor residualny $r_i = Ax_i - b$

2^o Metodę ϕ rozwiązujemy układ $Ad_i = r_i$ (7)

3^o Poprawiamy rozwiązanie $x_{i+1} = x_i - r_i$

Kryterium zakończenia powyższej procedury jest założona liczba iteracji lub "dokładność", tu rozumiana jako maksimum z wektora residualnego.

Mnożenie macierzowe $Y = A X$ (w (7) wykonujemy wg algorytmu:

$$Y_{p-1+i} = \sum_{\alpha=p}^{1} A_{(p-1+i-\alpha)(p+1)+\alpha} \, X_{p-1+i-\alpha} + \\ + \sum_{\alpha=o}^{p} A_{(p-1+i)(p+1)+\alpha} \, X_{p-1+i+\alpha} \qquad (8)$$

wymaga on około $n(2p+1)$ działań.

4. WYNIKI TESTÓW

Pakiet napisano w Turbo Pascalu, języku zyskującym coraz większe znaczenie w oprogramowaniu obliczeń inżynierskich. Skompilowanie pakietu do postaci modułu bibliotecznego (unit) umożliwia jego wygodne przyłączanie do dowolnego programu obliczeniowego. Przygotowano dwie wersje pakietu. Pierwsza dostosowana jest do przypadku, gdy cała macierz mieści się w RAM (dokładniej na stercie). Druga wersja jest naturalnym rozszerzeniem pierwszej i pozwala rozwiązywać układy większe pamiętane na dysku (macierz wczytywana jest do RAM kawałkami). Stosowanie wersji drugiej do układów mieszczących się w pamięci jest możliwe, ale nie jest ekonomiczne.

Wszystkie liczby (elementy macierzy i wektorów) pamiętane są w pojedyńczej precyzji, zaś wszystkie obliczenia prowadzone są z wykorzystaniem ich rozwinięć i arytmetyki podwójnej precyzji. (Istnieje też możliwość kompilacji bez koprocesora oraz kompilacji z podwyższoną precyzją - typ double+extended).

Wielkość zadań jakie można rozwiązać za pomocą prezentowanego pakietu ograniczona jest:

1^{o} Dla wersji pierwszej naturalnym ograniczeniem na rozmiar macierzy (n i p) jest wielkość dostępnej pamięci RAM.

2^{o} Dla wersji drugiej (dyskowej) ograniczona jest tylko szerokość pasma p (przy 400 kB na stercie $p \leqslant 220$), wymiar n teoretycznie może być dowolny, jednak w praktyce ogranicza go pojemność dysku i czas obliczeń.

Wykonano szereg testów pakietu (wybrane wyniki podane są w tabelach). Uzyskane rozwiązania udawało się zawsze poprawić iteracyjnie do zadowalającej dokładności. Już pierwszy krok procedury (7) poprawiał rozwiązanie, obniżając residuum o około 10^{-2}, następne kroki nie wnosiły istotnych poprawek, ich stosowanie nie wydaje się więc konieczne, co wydatnie skraca czas pracy potrzebny przy rozwiązywaniu układów równań.

Wersja 1 (układ mieści się w RAM)

n/p min:sek:100sek	1000/10	2000/20	3000/30
ROZKŁAD	0:18:84	2:06:99	6:43:76
ROZWIĄZYWANIE	0:05:33	0:20:50	0:44:32
1 ITERACJA	1:10:00	4:07:00	11:21:00
CAŁKOWITY	1:34:00	6:24:00	18:49:00
CZAS ZAPISU na dysk	0:47:51	2:02:48	4:24:08

Wersja 2 (układ pamiętany na dysku i wczytywany kawałkami)

n/p min:sek:100sek	$1000/_{10}$	$4000/_{40}$	$5000/_{50}$
ROZKŁAD	1:14:53	29:25:09	51:29:45
ROZWIĄZYWANIE	1:08:57	14:46:33	22:48:69
1 ITERACJA	2:10:00	24:15:00	37:03:00
CAŁKOWITY	4:33:00	68:27:00	111:81:00

DOKŁADNOŚĆ

(max. residuum)

n	1000	2000	3000	4000	5000
X_0	1.72 E-2	9.39 E-1	8.52 E-2	1.27 E-1	3.11 E-1
X_1	6.10 E-5	9.76 E-4	2.44 E-4	1.22 E-3	2.44 E-3

X_0 - rozwiązanie początkowe , X_1 - rozwiązanie po 1 iteracji

Większość czasu zużytego przy rozwiązywaniu układu pochłaniały transmisje dyskowe (szczególnie w wersji 2). Czasy te można kilkakrotnie zmniejszyć stosując lepszy sprzęt.

Omówione wyżej, przetestowane algorytmy można będzie w kolejnym etapie przepisać w języku C, tak aby nadawały się do współpracy z programami w Fortranie. Większość programów Systemu Koło napisana jest w Fortranie).

5. KONCEPCJA BUDOWY MODUŁU "ALGEBRY LINIOWEJ"

Jak wspomniano już we wstępie powstał pomysł budowy szerszego pakietu procedur wykorzystujących różne metody algebry liniowej dla macierzy pasmowych (np. wartości własne). Prawdopodobna jest bowiem ewolucja Systemu Koło w kierunku zagadnień dynamicznych i nieliniowych.

Okazuje się, że opracowane metody dostępu do elementów tablic stosować można wszędzie tam, gdzie pasmowy charakter macierzy wymusza specjalne zachowanie się algorytmu.

Przykładowo dla układów równań z macierzą niesymetryczną jednym z bardziej efektywnych algorytmów jest proces rozkładu **L U** Gaussa-Dolittle'a z częściowym wyborem elementu podstawowego w kolumnie.

Przy rozkładzie macierzy pasmowej **A** o paśmie q+1+p powstają pasmowe macierze trójkątne (z dokładnością do permutacji) **L** i **U**, a ich sumaryczne pasmo wynosi q+1+p+q=p+2q+1 (wzrasta o q) [4].

Przewidując ten efekt powiększyć należy pasmo macierzy **A** o q, tak aby elementy macierzy trójkątnych **L** i **U** tworzyć mogły się w miejsce elementów macierzy **A**. Wtedy proces rozkładu **L U** realizuje poniższy algorytm:

dla r = 1..n wykonaj kroki 1÷5:

1^{o} $D_{\alpha+(p+q)} := 0 \qquad \alpha = -(p+q) \,..\, -(p+1)$;

$D_{\alpha+(p+q)} := A_{(p+q-1+r+\alpha)w-\alpha+q} \qquad \alpha = -p..q$;

dla $\beta = (p+q)..1$ wykonaj krok 2 i 3:

2^{o} $A_{(p+q-1+r-\beta)w+\beta+q} := D_{P[r-\beta+(p+q)-1]-r+(p+q)}$;

$D_{(p+q)-\beta} := : D_{P[r-\beta+(p+q)-1]-r+(p+q)}$;

3^{o} $D_{(p+q)-\beta+\gamma} := D_{(p+q)-\beta+\gamma} +$

$- A_{(p+q-1+r-\beta+\gamma)w-\gamma} A_{(p+q-1+r-\beta)w+\beta} \qquad \gamma = 1..q$;

4^{o} $P[r+(p+q)-1] := r +$ (9)

indeks δ odpowiadający $\left(\max_{0 <= \delta <= q} D_{p+q+\delta} \right)$;

$A_{(p+q-1+r)w} := D_{(p+q+P[r+(p+q)-1]-r)}$;

$D_{(p+q)} := : D_{(p+q+P[r+(p+q)-1]-r)}$;

5^{o} $A_{(p+q-1+r+\gamma)w-\gamma} := D_{(p+q)} + \gamma / D_{(p+q)} \qquad \gamma = 1..q$;

gdzie $w = p+2q+1$,

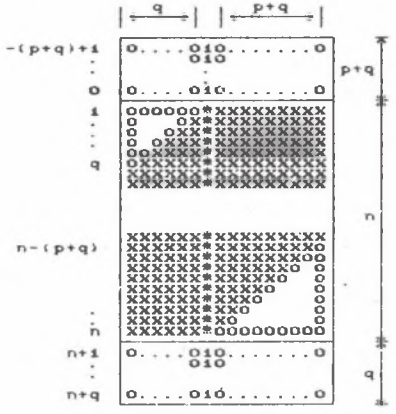
$A[0..(n+p+2q)w-1]$ wektor odpowiadający macierzy układu zgodnie ze wzorem (por. rys. 3):

$$a_{ij} = A_{(p+q-1+i)w+j-i+q}$$

$P[0..n+p+2q-1]$ wektor permutacji wierszy $(p_j = P[i+(p+q)-1])$ $D[0..p+2]$ wektor pomocniczy w podwójnej precyzji $(d_i = D_{i+(p+q)})$

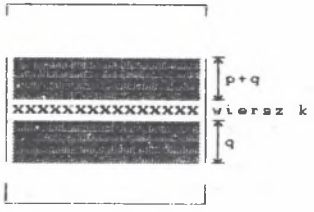
Algorytm (9) działa poprawnie gdy macierz układu powiększona jest o dodatkowe wiersze na początku i na końcu (rys. 3). Zauważyć można, że w k-tym kroku algorytmu potrzebne są do obliczeń tylko pewne wiersze macierzy (rys. 4), co podobnie jak w przypadku algorytmu (4) pozwala stosować go do bardzo dużych układów równań.

Trwają prace związane z oprogramowaniem i testowaniem przedstawionego niżej algorytmu.



Rys. 3. Budowa niesymetrycznej macierzy pasmowej

Fig. 3. Structure unsymmetrical band matrice



Rys. 4. Część macierzy wykorzystywana w k-tym kroku algorytmu

Fig. 4. Part of matrice used in k-step of algorithm

LITERATURA

[1] Bąk R., Mrówczyńska B., Matyja T.: Zastosowanie minikomputera IBM-PC do oceny wytrzymałości kolejowych zestawów kołowych, artykuł w tym zeszycie.

[2] Dryja M., Jankowscy J.M.: Przegląd metod i algorytmów numerycznych, WNT, Warszawa 1981.

[3] Ralston A.: Wstęp do analizy numerycznej, PWN, Warszawa 1975.

[4] Dahlquist G., Björck A.: Metody Numeryczne, PWN, Warszawa 1983.

Recenzent: Prof. dr hab. inż. Jan Broś

Wpłynęło do Redakcji 7.08.1989 r.

ПРОЦЕДУРЫ ГЕНЕРИРОВАНИЯ И РЕШЕНИЯ
БОЛЬШУХ СИСТЕМОВ ЛИНИЕЙНЫХ УРАВНИЕНИЙ
ДОПОЛНЯЮЩИЕ СИСТЕМУ ПРОГРАММ "КОЛО-ПЦ"

Р е з ю м е

В работе представлено пакет вычислителъных процедур полезных генерированию и решению болъших системов линейных уравниний с симетричными и ленточыми матрицами для минимашин типа IBM-PC.

Пакет становит дополнение системы программ нумерического анализа напряженного состаяния колесных пар. Делает возможным решение болъших задач метода коиечных элементов, которых матрицы ресткости болъше чем доступная операционная памятъ.

В пакете исполъзовано алгоритм Холесково и специалъные быстрые методы доступа к элэментом матрицы.

Авторы показали также идею постройки модулъя "линейной алгебры", которую можно применятъ в задачах методов конечных элементов и конечных разностей.

PROCEDURES OF GENERATING AND RESOLVING
A GREAT SYSTEM OF LINEAR EQUATIONS
COMPLEMENT SYSTEM OF KOLO-PC PROGRAMS

S u m m a r y

This paper presents set of procedures for generating and resolving a great system of linear equations whit symmetric and band matrices for IBM-PC.

This set is a complement for programs of Kolo-Pc System which is destined to stres analysis in wheel sets. These procedures resolve FEM problems, in which stiffness matrices exceed several times accessible RAM. There are used Cholesky's algorithm and special fast methods of the access to elements of the matrice.

The authors present a structure of "linear algebra" module and possibility of its application in FEM and FDT problems.