Manipulace s tridiagonálními maticemi

V 1D úloze jsou uzly přirozeně podél zkoumané úsečky – každý má jediného souseda vpravo a jediného vlevo. Tato vlastnost (neopakující se ve vyšších dimenzích) umožňuje očíslovat uzly (přirozeně) tak, že matice soustavy bude tridiagonální. Se vzniklou tridiagonální maticí

$$A = \begin{pmatrix} a_1 & c_1 & 0 & 0 & \dots & 0 \\ b_2 & a_2 & c_2 & 0 & \dots & 0 \\ \vdots & & & & & \\ 0 & \dots & 0 & 0 & b_N & a_N \end{pmatrix}$$

se dá jako s jedním z mála typů laborovat analyticky, například lze nezávisle na její velikosti pomocí zobecněného kontinuantu

$$K_0 = 1 \qquad K_1 = a_1 \qquad K_n = a_n K_{n-1} - b_{n-1} c_{n-1} K_{n-2}$$

explicitně (rekurzí) vyjádřit její determinant jako $\det A = K_N$.

Možnost spočítat v případě potřeby determinant libovolně velké matice rekurzivně (a bez rozvoje podél řádků/sloupců) představuje obrovské urychlení výpočtu.

Sám determinant tridiagonální matice nestačí pro řešení soustavy MKD, neboť dosazením pravé strany do jednotlivých sloupců bude tridiagonalita porušena. Přesto existuje způsob, založený na Gaussově eliminaci (která je obtížnosti $O(n^3)$), který tridiagonální soustavu umí rozřešit $O(n)$ - TDMA (Tridiagonal Matrix Algorithm). Tridiagonální soustavu

$$\begin{aligned} a_1 x_1 + c_1 x_2 &= d_1 \\ b_i x_{i-1} + a_i x_i + c_i x_{i+1} &= d_i \qquad i = 2, 3, \dots N-1 \\ b_N x_{N-1} + a_N x_N &= d_N \end{aligned}$$

můžeme převést na tvar

$$\begin{aligned} x_i + c'_i x_{i+1} &= d'_i \qquad i = 1, \dots N-1 \\ x_N &= d'_N \end{aligned}$$

kde

$$c'_1 = \frac{c_1}{a_1} \qquad c'_i = \frac{c_i}{a_i - c_{i-1} b_i}, i = 2, \dots N-1$$

$$d'_1 = \frac{d_1}{a_1} \qquad d'_i = \frac{d_i - d'_{i-1} b_i}{a_i - c'_{i-1} b_i}, i = 2, \dots N$$

Vzniklou soustavu již můžeme zpětným dosazováním snadno rekurzivně vyřešit:

$$x_N = d'_N, \qquad i = N-1, N-2, \dots 1 : \; x[i] = d'_i - c'_i x_{i+1}.$$

(podobně lze efektivně řešit i soustavy, které jsou jen blokově tridiagonální)

Ve speciálním případě digonálně konstantní matice $a_i = a$, $b_i = b$, $c_i = c$ lze psát explicitně i vlastní hodnoty této matice jako

$$\lambda_i = a + 2\sqrt{bc} \cos\left(\frac{\mathrm{i}\pi}{N+1}\right),$$

explicitně lze v tomto případě nalézt také vlastní vektory.

Vlastní hodnoty ve více dimenzích

Při přechodu k vícerozměrným úlohám již není možné uzly číslovat vhodně organizovaným způsobem, zejména ne v případě MKP. Získané matice tak již nebudou tridiagonální a pro nalezení jejich vlastních hodnot musíme použít obecnější algoritmy.
V obecném případě pro nalezení vlastních hodnot energie z $H - EI$ použijeme iterativní QR algoritmus: ($Q^{\mathrm{T}}Q = I$ je ortogonální matice, R je horní trojúhelníková matice)

1) nechť $H_0 \equiv H$
2) nalezneme QR dekompozici $H_0 = Q_0 R_0$.
3) určíme $H_{k+1} = R_k Q_k$.
4) loop 2) s H_{k+1}

Dostatečným počtem kroků postup iteruje k trojúhelníkové matici podobné s výchozí, takže hodnoty z diagonály jsou přímo vlastní hodnoty řešeného problému.
Vzhledem ke své povaze může být QR rozkald použít také přímo pro solver lineárních soustav.

QR dekompozice s využitím Householderovy transformace

Uvažujme matici $H = (h_{ij})$. QR dekompozice s využitím Householderovy transformace spočívá v následujícím schematu:

Householderova transformace v roli optimalizace
(získáme horní Hessenbergovu matici, obtížnost $(10/3)n^3 + O(n^2)$)
QR dekompozice
(rozklad na ortogonální a horní trojúhelníkovou matice, obtížnost $6n^2 + O(n)$
pro horní Hessenbergerovu matici)

Celkový algoritmus:

1) vezměme první sloupec H, h_{1j} s normou $|h_{1j}|$

2) vypočteme vektor $u_j = h_{1j} - |h_{1j}|e_1$,
kde $e_1 = (1, 0, 0, \ldots)$

3) spočteme matici $Q[1]_{ij} = I_{ij} - \frac{2}{|u_j|^2} u_i u_j$

Potom $H[1] = Q[1]H$ má v prvním sloupci pod hlavní diagonálou nuly a celý proces můžeme opakovat iterativně s tím, že se v každém dalším kole zaměříme pouze na zbývající největší minor.

Přitom pro násobení $Q[k]H[k-1]$ musíme $Q[k]$ vždy doplnit shora příslušnou částí jednotkové matice.

Na závěr tvoří $Q = Q[1]Q[2]\ldots Q[K]$ a $R = Q^T H$ hledanou QR dekompozici $H = QR$.

V případě Hermiteovské matice jsou vznikající Hessenbergovy matice tridiagonální, a celý algoritmus Hauseholderovy transformace lze dále optimalizovat. Samotnou otázku tvoří problematika konvergence (pomalá, jsou-li vlastní hodnoty blízko); řeší se zaváděním posunů spektra.

Ukázka kódu pro výpočet QR algoritmu (perl)

```
for ($cyklus=1;$cyklus<=100;$cyklus++){$kolo=1;

for ($i=$kolo;$i<=$dim;$i++){for ($j=$kolo;$j<=$dim;$j++){$fullQ[$i][$j]=0}};
for ($i=$kolo;$i<=$dim;$i++){$fullQ[$i][$i]+=1};

for ($kolo=1;$kolo<=($dim-1);$kolo++){

   $norma=0;
   for ($j=$kolo;$j<=$dim;$j++){$u[$j]=$matice[$kolo][$j];$norma+=$u[$j]*$u[$j]};
   $u[$kolo]-=sqrt($norma);

   $knorma=0;
   for ($j=$kolo;$j<=$dim;$j++){$knorma+=$u[$j]*$u[$j]};

   for ($i=$kolo;$i<=$dim;$i++){for ($j=$kolo;$j<=$dim;$j++){
      $Q[$i][$j]=-2*$u[$i]*$u[$j]/$knorma}};

   for ($i=$kolo;$i<=$dim;$i++){$Q[$i][$i]+=1};

   for ($i=$kolo;$i<=$dim;$i++){for ($j=1;$j<=$dim;$j++){$nfullQ[$i][$j]=0;
      for ($k=$kolo;$k<=$dim;$k++){$nfullQ[$i][$j]+=$fullQ[$k][$j]*$Q[$k][$i]}}};

   for ($i=$kolo;$i<=$dim;$i++){for ($j=$kolo;$j<=$dim;$j++){$nmatice[$i][$j]=0;
      for ($k=$kolo;$k<=$dim;$k++){$nmatice[$i][$j]+=$Q[$k][$j]*$matice[$i][$k]}};

   for ($j=1;$j<=$dim;$j++){for ($i=1;$i<=$dim;$i++){
      $matice[$i][$j]=$nmatice[$i][$j];$fullQ[$i][$j]=$nfullQ[$i][$j]}};
}

print "\n";

for ($i=1;$i<=$dim;$i++){for ($j=1;$j<=$dim;$j++){$puvodni[$i][$j]=0;
   for ($k=1;$k<=$dim;$k++){$puvodni[$i][$j]+=$nmatice[$k][$j]*$fullQ[$i][$k]};
      $matice[$i][$j]=$puvodni[$i][$j]}};

for ($j=1;$j<=$dim;$j++){for ($i=1;$i<=$dim;$i++){
   printf "%1.6f ", $puvodni[$i][$j] };print "\n";}
}
```