UŽITEČNÉ FUNKCE PRO MATLAB

Miroslav BALDA

Ústav termomechaniky AVČR, Centrum diagnostiky materiálů

1. Úvod

Při dlouhodobém využívání jazyka **MATLAB** zjistíme, že přes jeho dokonalost by bylo dobré jisté funkce doplnit ve prospěch uživatele a jeho potřeb. Po delším váhání předkládám k obecnému použití dvě skupiny funkcí které mohou usnadnit a často i výrazně zrychlit práci uživatele. První skupinu, zařazenou do adresáře **SubML**, tvoří funkce obecného zaměření, z větší části pak pro vstupní a výstupní operace. Ve druhé skupině umístěné v adresáři **Arr3D** jsou funkce pro práci s trojrozměrnými poli a maticovými časovými řadami.

Dále jsou obě výše uvedené skupiny popsány podrobněji. Všechny M-funkce jsou zapsány prostředky jazyka **MATLAB** verze 5.3 a měly by tudíž pracovat v jakémkoliv prostředí, ve kterém je tento produkt této nebo vyšší verze nainstalován.

2. Obecné procedury

V adresáři **SubML** je několik funkcí, které lze s výhodou použít při tvorbě vlastních aplikačních programů. Většina z nich se týká vstupu a výstupu dat. Jsou to následující M-funkce:

INP - Vstup dat z klávesnice

Funkce slouží k operativnímu zadávání dat z klávesnice, ale na rozdíl od standardní funkce **input** umožňuje zadávat implicitní hodnoty zvolené uživatelem. Její užití je výhodné ve všech případech, kdy se užívá funkce **input**. Je to zejména při parametrických studiích a při zkouškách programů, kdy je zapotřebí parametry běhu programu operativně měnit. Uživatel ocení tuto funkci i v případě, že chce užít program s odstupem času a pro jeho vyzkoušení již neví jaké vhodné parametry programu zadat. Volání může mít tvar:

`x = inp;` jako funkce **input** s výzvou „`input = `“

`x = inp(výzva);`
: podobná funkci **input**; `výzva je řetěz`!

`x = inp(výzva,hodnota);`
: vstup na výzvu `výzva` s doporučenou hodnotou `hodnota`, která může být číselná nebo řetěz. Za ní vystoupí znaky „=>“, na které odpovíme buď ENTER při přijetí nabídky, anebo vložením nové hodnoty. Je-li nabídnutou hodnotou řetěz, je i výsledná hodnota řetězem. Pokud se vkládá nový, pak již bez apostrofů,

`x = inp(výzva,hodnota,formát);`
: `formát` je standardním formátem **MATLAB**u, který bude zachován i pro další vstupy do procedury **inp**, pokud nebude novým formátem změněn. Implicitní formát je '%9.4f',

`x = inp(výzva,hodnota,formát,mezer);`
: parametr `mezer` je celé číslo udávající, o kolik mezer má být `výzva` posunuta vpravo od okraje obrazovky a tím i zvýrazněna. Implicitně je nastavena na 10. Nově nastavená hodnota se zachovává i pro další vstupy.

OUT - Formátový výstup argumentu do souboru

Jde o operativnější variantu standardní funkce **printf**, pomocí které lze formátovat numerická i znaková data na obrazovku a případně i do souboru.

`out(arg)` výstup argumentu `arg` na obrazovku podle jeho povahy, t. j. buď jako řetěz znaků nebo jako číslo ve formátu '%9.4f'.

`out(arg,formát)`
: výstup argumentu na obrazovku v požadovaném formátu. Parametr `formát` je buď ve standardní formě, anebo lze použít i zkrácený zápis 'E' pro formát '%11.3e'. Argument může být i komplexní,

`out(arg,formát,file)`
: formátový výstup argumentu do souboru o jménu, které je obsahem posledního argumentu (jako řetěz)

NCHAR - Výstup n stejných znaků

Někdy je zapotřebí pro vhodnou úpravu vystupujícího dokumentu vytisknout určitý počet stejných znaků. To lze zajistit voláním:

`nchar(znak,počet)`
: vystoupí žádaný počet znaků `znak` na nové řádce,

`nchar(znak,počet,koncový znak)`
: dtto s přidaným koncovým znakem (např. '\n' pro přechod na novou řádku).

CHECK - Podmíněný výstup prvků matice na obrazovku

Slouží pro rychlou kontrolu prvků reálné nebo komplexní matice splňujících určitou podmínku. Jeho použití lze očekávat zejména v etapách ladění, kdy uživatele zajímají jen některé hodnoty matic, aniž ví dopředu, které a v jakých pozicích se mohou vyskytnout. Funkce **check** se volá některým ze způsobů:

`check(matice)`
: výstup celé matice po sloupcích, každý prvek na nové řádce ve tvaru
`index řádky i  index sloupce j  hodnota prvku o indexech (i,j),`
ve formátu '%13.4e'

`check(matice,podmínka)`
: výstup pouze prvků splňujících podmínku zadanou v druhém argumentu ve tvaru řetězu, který obsahuje logický výraz např.: `check(A,'A>.5')`,

`check(matice,podmínka,formát)`
: stejný výstup jako v předešlém případě, avšak v požadovaném formátu.

Podmínka se zapisuje ve tvaru řetězu jako logický výraz vázaný pouze na neindexovanou proměnnou z prvního parametru. Žádný jiný podřetěz podmínky nesmí obsahovat jméno proměnné.

SPRINTFM - Formátová konverze reálné matice

Pokud se funkce **sprintfm** použije bez středníku, plní podobnou funkci jako standardní funkce **disp**. Na rozdíl od ní může uživatel zadat i formát pro konverzi reálné matice. Jde tedy o zjednodušenou maticovou analogii standardní funkce **sprintf**, ale na rozdíl od ní se matice tiskne po řádkách. Volá se:

`sprintfm(matice)`
: výstup `matice` na obrazovku ve formátu '%7.3f',

`sprintfm(formát,matice)`
: dtto v požadovaném formátu,

`s = sprintfm(matice);`
: konverze `matice` do řetězu ve formátu '%7.3f',

`s = sprintfm(formát,matice);`
: dtto v požadovaném formátu.

Příklad: Sestav matici řádu 10 celých dvouciferných pseudonáhodných čícel a vytiskni ji:

```
c = sprintfm('%3i',fix(100*rand(10)))
```

Vypočti tabulku funkce sinus pro sloupcový vektor `arg` a vytiskni ji

```
sprintfm('%5.1f %13.10f',[arg, sin(arg)])
```

LOADMX - Čtení požadovaného souboru do matice

Při potřebě zpracovávat datové soubory, jejichž jména v době přípravy programu nejsou známá, je výhodné použít funkci **loadmx**, která obsah znakového souboru - prvky matice - pošle do matice požadovaného jména. Proti obyčejné funkci **load** má výhodu v tom, že výsledná matice má jméno, které zadá uživatel, a se kterým lze dále pracovat v programu, a ne jméno souboru, který mohl vzniknout bez vazby na daný program. Vyvolání funkce **loadmx** se uskuteční příkazem:

`matice = loadmx(soubor);`
: Znakový soubor, jehož jméno je uložené jako řetěz v argumentu funkce **loadmx**, se přečte a výsledek se uloží do požadované `matice`.

Příklad:

```
xy = loadmx(inp('matice dat','data.txt'));
```

DELIJ - Vypuštění požadovaných řádek a sloupců matice

Tato funkce se zajistí např. voláním

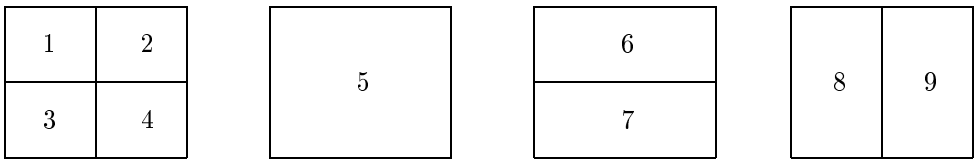
```
B = delij(A,ia,ja);
```

kde `A` je původní matice,
`B` je matice vzniklá z `A` vypuštěním řádek a sloupců, jejichž indexy jsou uloženy ve vektorech `ia` a `ja`. Není na závadu, je-li některý z vektorů prázdnou maticí `[]`.

FIG - Umístění obrázku na zvoleném místě obrazovky

Standardní umístění obrázku na obrazovce, jak ho nabízí **MATLAB**, nemusí být vždy vhodné. Uživatel má sice možnost ho změnit i implicitně např. již v M-funkci **startup** pomocí příkazu **set**, ale v případě, že by jich mělo být více, vznikají jisté problémy. Těm lze zčásti zabránit volbou určitých poloh obrázků na obrazovce. Výsledek umisťování by však mohl záviset na rozlišitelnosti obrazovky. Proto byla sestavena funkce **fig**, která tyto potíže omezuje.

Ať poloha obrázku je charakterizována jeho pevným indexem `n`, který lze vyčíst ze schématu uvedeného na následujícím obrázku:



Obrázek v požadované poloze se vytvoří i s příslušným identifikátorem `fih` obrázku příkazem

```
fih = fig(n);
```

kde `n` je index obrázku.

Tímto voláním se umístí na požadované pozici prázdný rámec obrázku bez nástrojové lišty a do `fih` se umístí jeho identifikační číslo (handle) rovné pořadí volání `fih`. Zde je třeba upozornit, že index polohy a identifikační číslo obrázku jsou na sobě nezávislé.

Při zpracování simultánního měření z velkého počtu měřících míst bývají data z jednotlivých měřících míst v samostatných souborech s mnemotechnickými jmény a se specifickým rozšířením (např. číslem měření). Navíc tyto soubory mívají stejnou strukturu obsahující několik řádek úvodních informací následovaných vlastními daty, jichž bývá ve všech souborech stejný počet. Pokud se tato data zpracovávají stejným způsobem, je vhodné použít maticový přístup zpracování. K přípravě a sestavení matice slouží následující dvě funkce:

SETLIST - Seznam jmen vybraných souborů ve zvoleném adresáři

Funkce **setlist** vytvoří seznam jmen souborů se stejnou extenzí, které jsou umístěné v požadovaném adresáři. Volá se

```
[list,m] = setlist(pth,ext)
```

`pth` je definice cesty k požadovanému adresáři s datovými soubory,
`ext` je rozšíření jména souborů (extension), které mají být vybrány,
`list` je seznam souborů v požadovaném adresáři a se žádaným rozšířením,
`m` je počet jmen v seznamu.

GETFILES - Čti soubory podle seznamu a vytvoř matici

Touto funkcí zajistíme přečtení potřebného počtu dat z určitého adresáře ze souborů uvedených v seznamu vytvořeném funkcí **setlist** tak, že každý soubor vytvoří jeden sloupec matice dat. Její volání je

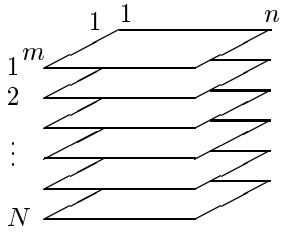
```
[A,N] = getfiles(pth,list,skipl,N)
```

`pth` je cesta k adresáři s požadovanými daty stejná jako u **setlist**,
`list` je seznam jmen souborů vytvořený funkcí **setlist**, `skipl` je počet úvodních řádek souborů, které se při čtení přeskočí,
`N` je dimenze sloupcového vektoru výsledné matice (počet dat, které se ze souboru přečtou), na vstupu žádaná, na výstupu dosažená a
`A` je výsledná matice dat

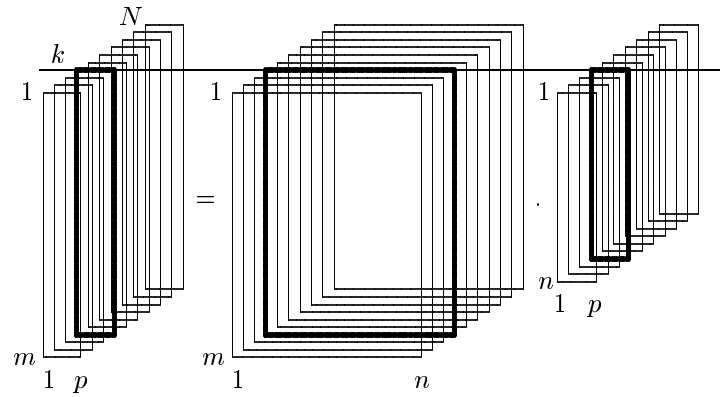
3. Funkce pro práce s trojrozměrnými poli

Několik posledních verzí **MATLAB**u umí ukládat data i do vícerozměrných polí. Tato skutečnost je velmi vítaná zejména tam, kde se vyskytují úlohy, při nichž vznikají řady matic stejného typu. Práce s vícerozměrnými poli však není příliš jednoduchá, pokud chceme s nimi pracoval zcela obecně. Brzy totiž zjistíme, že kromě funkcí pro generování matic, jako jsou **rand, randn, ones** a **zeros**, a operací nad prvky polí jako jsou slučování, „tečkové“ operace, prvkové funkce a funkce pracující nad sloupci, nejsou běžné operace, jako jsou násobení, (pseudo)inverse, transpozice a funkce matic , definovány. V tom případě si je uživatel musí udělat sám za použití cyklů. Aby nemusel vymýšlet vždy znovu efektivní realizaci potřebné funkce, byla sestavena knihovna procedur uložená v adresáři **Arr3D** pro nejčastěji se vyskytující operace. Ta rovněž obsahuje dvě skupiny M-funkcí. První z nich pracuje s třírozměrnými poli a má proto v názvu číslici 3. Druhá skupina je určena pro práce s maticovými časovými nebo frekvenčními řadami uloženými v obyčejných maticích.

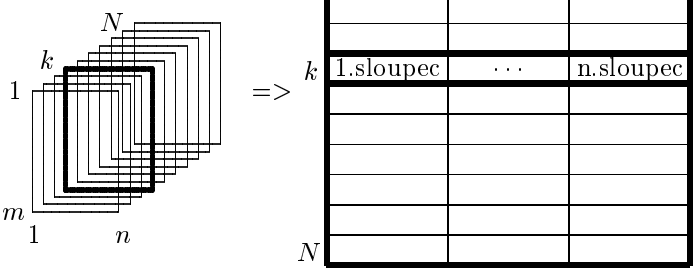
Pro pole `A` definované s rozměry `(N,m,n)` podle schénatu 1 lze velmi snadno použít „sloupcové“ maticové funkce (**fft, mean, std**), u nichž je funkce aplikována přes první rozměr. Výsledné pole je typu `(m,n)`. Rovněž lze použít všechny prvkové maticové funkce (`sin(A), exp(A),...`), u nichž je výsledné pole stejného typu jako argument. Totéž platí pro slučování a prvkové (tečkové) aritmetické operace např. ve tvaru `c = a+b` nebo `c = a.*b`. Bohužel totéž nelze provést např. pro násobení matic z trojrozměrného pole. Nejen, že *nelze* napsat `c = a*b`, ale ani po rozvinutí do cyklu `for k=1:n, c(k,:,:) = a(k,:,:)*b(k,:,:); end` dostaneme pouze chybové hlášení, ze kterého se dozvíme, že tato operace není implementována, a to i v případě, že by matice `a(k,:,:)` a `b(k,:,:)` byly formálně násobitelné.



Naproti tomu pokud by `a(:,:,k)` a `b(:,:,k)` byly násobitelné, **MATLAB** je správně pronásobí. Toho lze využít pro násobení polí ukládaných podle schématu 2 uvedeného ve vedlejším obrázku. Dílčí matice pole jsou rozlišeny třetím indexem, přičemž první dva indexy odpovídají počtu jejich řádek resp. sloupců. Tento způsob ukládání je použit u dále uvedených funkcí, jejichž název končí číslicí 3. Bohužel tento tvar není vhodný pro výše uvedené funkce **fft, mean, std**.



Z tohoto důvodu se jako lepší ukázal třetí způsob ukládání trojrozměrných polí. Jeho použití je vhodné zejména pro práce spojené se zpracováním maticových časových (frekvenčních) řad. V tomto formátu se každá maticová časová řada o délce `N` hodnot v každém prvku uloží do obyčejné matice o `N` řádcích a `m.n` sloupcích, přičemž `k`-tá řádka této matice je tvořena sloupci `k`-té matice maticové časové řady podle schématu 3. Toto schéma bylo využito u funkcí, jejichž název začíná písmeny `Ts` (Time series). Obě skupiny funkcí jsou umístěny v adresáři **Arr3D**.



3.1 Funkce pro trojrozměrná pole

Jsou určeny pro práci s poli podle druhého schématu, tedy s dimenzemi `(m,n,N)`. Jde tedy o množiny N matic typu (m, n). Díky tomuto uspořádání lze nad maticemi provádět běžné operace bez velikých úprav. Z aritmetických operací nejsou vypracovány funkce pro slučování, které jako prvkové operace se zrealizují přímo zápisem příkazu např. ve tvaru `C = A+B`. Další běžné operace se vyvolávají pomocí funkcí zajišťujících násobení dílčích matic, řešení systémů lineárních algebraických rovnic, transpozice atd.:

MUL3 - Násobení matic z trojrozměrných polí

Násobení trojrozměrných polí ve formátu 2 popisuje rovnice

$$\boldsymbol{C}_k = \boldsymbol{A}_k \, \boldsymbol{B}_k \,, \qquad 1 \leq k \leq N \,, \tag{1}$$

Získá se voláním funkce **mul3** pomocí příkazu

`C = mul3(A,B);`
kde `A` je pole typu `(m,n,N)` a `B` typu `(n,p,N)`. Výsledné pole `C` bude typu `(m,p,N)`.

Analogií dělení u skalárů je řešení systémů lineárních algebraických rovnic ve smyslu nejmenších čtverců u matic `A` a `B`:

LSQL3 - Řesení N systémů lineárních algebraických rovnic

Procedura **lsql3** je určena pro řešení problému

$$\boldsymbol{C}_k = \boldsymbol{A}_k^{+} \, \boldsymbol{B}_k \,, \qquad 1 \leq k \leq N \,, \tag{2}$$

kde symbol "+" v pozici exponentu u matice $\boldsymbol{A}_k$ značí její Moore-Penroseovu pseudoinverzi. Problém se vyřeší příkazem

`C = lsql3(A,B);`
kde `A` je pole typu `(m,n,N)` a `B` je typu `(m,p,N)`. Výsledné pole `C` je typu `(n,p,N)`.

LSQR3 - Řesení N systémů lineárních algebraických rovnic

Procedurou **lsqr3** se řeší problém s pseudoinvertovanou maticí stojící zprava

$$\boldsymbol{C}_k = \boldsymbol{A}_k \, \boldsymbol{B}_k^{+} \,, \qquad 1 \leq k \leq N \tag{3}$$

Na rozdíl od funkce `lsql3` se nyní pseudoinvertuje matice `B`. Volání funkce `lsqr3` může mít tvar

`C = lsqr3(A,B);`
kde pole `A` je typu `(m,n,N)` a `B` je typu `(p,n,N)`. Výsledné pole `C` je typu `(m,p,N)`.

INV3 - Inverze matic z trojrozměrných polí

Méně častou, nicméně potřebnou, je funkce **inv3**, která zinvertuje N matic řádu m, z nichž je pole složeno. Tuto úlohu vyjadřuje rovnice

$$\boldsymbol{C}_k = \boldsymbol{A}_k^{-1} \,, \qquad 1 \leq k \leq N \tag{4}$$

Vyvolá se příkazem

`C = inv3(A);` kde `A` i `C` jsou pole typu `(m,m,N)`.

PINV3 - Pseudoinverze matic trojrozměrných polí

Funkce **pinv3** je určena k pseudoinvertování matic, znichž je pole složeno, tedy k výpočtu

$$\boldsymbol{C}_k = \boldsymbol{A}_k^{+} \,, \qquad 1 \leq k \leq N \tag{5}$$

Zajistí se voláním

`C = pinv3(A);` kde pole `A` je typu `(m,n,N)` a pole `C` je typu `(n,m,N)`.

NORM3 - Normy matic trojrozměrných polí

Funkce **norm3** počítá vektor $\boldsymbol{c}$ norem matic

$$\boldsymbol{c}_k = \| \, \boldsymbol{A}_k \, \|_p, \qquad 1 \leq k \leq N \,, \tag{6}$$

kde p je příznak typu normy. Vyvolá se příkazem

`c = norm3(A);`

kde `A` je pole typu `(m,n,N)` a `c` je řádkový vektor $\boldsymbol{c}$ dimenze N. Skládá se z prvků rovných maximálním singulárním číslům matic $\boldsymbol{A}_k$, tedy max(svd($\boldsymbol{A}_k$))

`c = norm3(A,par);`

kde `par` je parametr stejný jako u standardní funkce **norm** pro matice

1	pro $c_k = \max_j \sum_i \mid a_i \mid_{jk}$, tedy největší ze sumy sloupců matice $\boldsymbol{A}_k$,
2	pro c_k stejné jako při volání `norm3(A)`, tj. max(svd($\boldsymbol{A}_k$))
`inf`	pro $c_k = \max_i \sum_j \mid a_j \mid_{ik}$, tedy největší z řádkových sum matice $\boldsymbol{A}_k$,
`'fro'`	pro $c_k = \sqrt{\sum_{i,j} \boldsymbol{A}_k^T \boldsymbol{A}_k}$

TRN3 - Transpozice matic trojrozměrného pole

Funkce **trn3** slouží k transpozicím dílčích matic podle vztahu

$$\boldsymbol{C}_k = \boldsymbol{A}_k^T \qquad 1 \le k \le N \tag{7}$$

Její volání je

`C = trn3(A);` kde `A` je pole typu `(m,n,N)` a `C` pole typu `(n,m,N)`

DOWN3 - Přeskupení matic z pole do obyčejné matice

V praxi mohou nastat případy, kdy je výhodné pracovat s maticí $\boldsymbol{B} = [\boldsymbol{A}_1^T,\ \boldsymbol{A}_2^T,\ \cdots\ ,\ \boldsymbol{A}_N^T,]^T$ místo s polem $\boldsymbol{A}$. V matici $\boldsymbol{B}$ jsou dílčí matice $\boldsymbol{A}_k$ uspořádány „pod sebou“. K této operaci slouží funkce **down3**, která se volá:

`B = down3(A);` kde `A` je pole typu `(m,n,N)` a `B` matice typu `(mN,n)`.

3.2 Funkce pro časové řady

Již v úvodu této kapitoly bylo řečeno, že pro zpracování časových nebo frekvenčních řad, kde se často vyskytují i operace s Fourierovou transformací a výpočty středních hodnot a směrodatných odchylek, je výhodnější ukládat jednotlivé vzorky časových řad do sloupců obyčejné matice. K tomu bylo užito třetí schéma. Jména funkcí využívajících tento způsob ukládání začínají písmeny **ts**. Tyto funkce až na malé odchylky plní stejné úlohy jako funkce z předešlého odstavce, ale s ohledem na jednodušší strukturu matic vyžadují jistou modifikaci parametrů při volání. I když se dále bude mluvit jen o časových řadách, funkce lze pochopitelně užít i pro frekvenční řady.

TSMUL - Násobení maticových časových řad

Touto funkcí se realizuje řada velmi významných operací. Jako příklad může sloužit výpočet Fourierova obrazu odezev $\boldsymbol{y}(f)$ diskrétního systému s mnoha vstupy a mnoha výstupy (MIMO), jako součin pole matic frekvenčních odezev $\boldsymbol{G}(f)$ vektorem Fourierových obrazů časových řad buzení $\boldsymbol{x}(f)$.

$$\boldsymbol{y}(f) = \boldsymbol{G}(f)\,\boldsymbol{x}(f)\,, \tag{8}$$

Součin dílčích matic se v tomto případě zajistí pomocí funkce **tsmul** voláním

`y = tsmul(G,x,mG);`

`G` je pole typu `(N,mG.nG)`,
`x` je pole typu `(N,nG.nx)`, a kde
`mG` je počet řádek dílčích matic $\boldsymbol{G}(f_k)$ - výstupů, přičemž
`nG` je počet sloupců těchto matic = vstupů a
`nx` je počet realizací buzení

TSMULC - Násobení maticové časové řady obyčejnou konstantní maticí

Funkce **tsmulc** na rozdíl od **tsmul** pronásobí všechny dílčí matice $\boldsymbol{A}_k$ maticové časové řady stejnou obyčejnou maticí $\boldsymbol{B}$ podle vztahu

$$\boldsymbol{C}_k = \boldsymbol{A}_k\,\boldsymbol{B} \qquad \text{nebo} \qquad \boldsymbol{C}_k = \boldsymbol{B}\,\boldsymbol{A}_k \tag{9}$$

za předpokladu, že matice $\boldsymbol{A}_k$ a $\boldsymbol{B}$ jsou násobitelné. Tato operace se vyvolá příkazem

`C = tsmulc(A,B)` nebo `C = tsmulc(B,A)`

`A` je maticová časová řada typu `(N,mAk.nAk)`,
`B` je maticová časová řada typu `(N,mB.nB)` a
a `mAk, NAk, MB, nB` jsou takové dimenze, aby matice byly násobitelné.

TSLSQL - Řešení problému nejmenších čverců časových řad

Funkce **tslsql** vypočítá řešení úlohy popsané rovnicí (2), ovšem s poli uloženými ve tvaru obyčejných matic podle schématu 3. To se získá voláním

`C = tslsql(A,B,mAk);`

`A` je matice typu `(N,mAk.nAk)`,
`B` je matice typu `(N,mAk.nBk)`,
`C` je matice typu `(N,nAk.nBk)`, přičemž
`mAk` je počet řádek dílčích matic $\boldsymbol{A}_k$,
`nAk` je počet sloupců dílčích matic $\boldsymbol{A}_k$ a
`nBk` je počet sloupců dílčích matic $\boldsymbol{B}_k$

TSLSQR - Řešení problému nejmenších čverců časových řad

Tato funkce řeší podobnou úlohu jako funkce **lsql3** popsanou rovnicí (3). Vyvolá se příkazem

`C = tslsqr(A,B,mAk);`

`A` je matice typu `(N,mAk.nAk)`,
`B` je matice typu `(N,mBk.nAk)` a
`C` je matice typu `(N,mAk.mBk)`, přičemž
`mAk` je počet řádek dílčích matic $\boldsymbol{A}_k$,
`nAk` je počet sloupců dílčích matic $\boldsymbol{A}_k$ a
`nBk` je počet sloupců dílčích matic $\boldsymbol{B}_k$

Příklad:

Ať $\boldsymbol{X}(f_n)$ je maticová frekvenční řada diskrétních Fourierových obrazů nezávislých buzení v n místech diskrétního lineárního dynamického systému a $\boldsymbol{Y}(f_n)$ jim odpovídající matice stejných obrazů odezev v m místech systému. Potom z rovnice (8) lze najít maticovou řadu frekvenčních přenosů ve tvaru $\boldsymbol{G}(f_n) = \boldsymbol{Y}(f_n)\,\boldsymbol{X}(f_n)^+$, což lze zapsat příkazem

```
Gf = tslsqr(Yf,Xf,m);
```

TSINV - Inverze maticové časové řady

Funkce **tsinv** invertuje všechny matice řádu m maticové časové řady $\boldsymbol{A}$ po vyvolání příkazu

`C = tsinv(A);` `A` je matice typu `(N,mAk`^{2}`)` a `mAk` je řád matic $\boldsymbol{A}_k$.

TSTRN - Transpozice matic časové řady

Funkce ztransponuje dílčí matice časové řady příkazem

`C = tstrn(A,mAk):`

`A` je matice typu `(N,mAk,mAk)` a
`mAk` je řád dílčích matic $\boldsymbol{A}_k$.

TSDOWN - Přeskupení dílčích matic z časové řady do submatic

Funkce **tsdown** přetransformuje maticovou časovou řadu $\boldsymbol{A}$ do jiné matice $\boldsymbol{B}$ tak, že dílčí matice $\boldsymbol{A}_k$ budou jejími submaticemi, t. j. $\boldsymbol{B} = [\boldsymbol{A}_1^T,\ \boldsymbol{A}_2^T, \cdots,\ \boldsymbol{A}_N^T]^T$

`B = tsdown(A,mAk);`

`A` je matice typu`(N,mAk.nAk)`,
`B` je matice typu`(N.mAk,nAk)`,
`mAk` je počet řádek dílčích matic $\boldsymbol{A}_k$ a
`nAk` je počet sloupců dílčích matic $\boldsymbol{A}_k$.

Příklady:

```
A = rand(4,6); % Maticová časová řada se 4 čtvercovými maticemi řádu 3.
I = tsdown(tsmul(tsinv(A),A,3),3); % I je matice složená ze 4 jednotkových submatic řádu 3.
```

TSDIAG - Výběr hlavní diagonály z maticové časové řady

Touto funkcí lze z maticové časové řady vyjmout diagonálu a vložit jí do časové řady, jejíž dílčí matice budou obsahovat pouze diagonální prvky z jim odpovídajících původních matic, zatímco ostatní budou nulové. její volání je následující:

```
D = tsdiag(A);
```
pro `A` se čtvercovými dílčími maticemi,

```
D = tsdiag(A,mAk)
```
pro `A` typu `(N,mAk.nAk)`, kde `N` je délka maticové časové řady.

V adresáři **Arr3D** jsou ještě další funkce umožňující zpětmou transformaci matice obsahující dílčí submatice maticové časové řady uspořádané „pod sebou“ do tvaru trojrozměrného pole nebo maticové časové řady:

DOWN2ARR - Zpětná transformace matice ve tvaru sloupce submatic do pole

Jde o inverzní funkci k **down3**, která se vyvolá příkazem

```
B = down2arr(A,mAk);
```
- `A` je matice typu `(mAk.N,nAk)` složená z dílčích submatic $\boldsymbol{A}_k$ původně trojrozměrného pole,
- `mAk` je počet řádek a `nAk` počet sloupců submatice $\boldsymbol{A}_k$ a
- `B` je trojrozměrné pole podle schématu 2.

DOWN2TS - Zpětná transformace matice ze sloupce submatic do maticové časové řady

Funkce **down2ts** je inverzní funkcí k **tsdown** a volá se

```
B = down2ts(A,mAk);
```
- `A` je matice typu `(mAk.N,nAk)` složená z dílčích submatic $\boldsymbol{A}_k$ původně trojrozměrného pole,
- `mAk` je počet řádek a `nAk` počet sloupců submatice $\boldsymbol{A}_k$ a
- `B` je matice maticové časové řady podle schématu 2.

4. Závěr

V tomto příspěvku předkládá autor řadu drobných funkcí, které se mu v průběhu let osvědčily při tvorbě aplikačních programů. Funkce jsou uspořádány do dvou adresářů, z nichž první, **SubML**, obsahuje funkce obecného užití, kdežto druhý, **Arr3D**, funkce pro práci s trojrozměrnými poli případně maticovými časovými řadami. Mohou najít uplatnění při zpracování měření na dynamických systémech, kde vznikají data tohoto charakteru.

Oba adresáře lze nalézt na www stránkách firmy Humusoft a přidat mezi prohledávané pomocí standardní funkce **addpath**.

Poděkování

Příspěvek vznikl v rámci prací na projektu č. 101/99/0103 za podpory Grantové agentury ČR.

Prof. Ing. Miroslav BALDA, DrSc., Ústav termomechaniky AVČR, Veleslavínova 11, 301 14 Plzeň
tel.: 019-7236415 balda@cdm.it.cas.cz