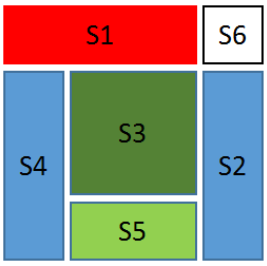
Lab 4: Basic matrix calculations

Save all the exercises to a single MATLAB script (m-file) with the name: `lab4_XXX.m` (Where you replace XXX with your surname). Use cell mode (%%) to separate the exercises. Remember: function definitions must go to the end of the file. **You cannot use any `for/while` or any other form of cycle while implementing the assignments. The use of the built-ins: `horzcat, vertcat, transpose` is also prohibited, use the matrix operators instead. If you work in pairs, put both of your names in the comment on the first line of each submitted file.**

Exercise 1: Matrix indexing

Using range indexing, cut the matrix **A** of size 4x4 into submatrices as shown in the picture below and then transpose **A** per partes – by transposing and repositioning the pieces S1 – S6 into an all-zero matrix **B** of the same size as **A**. Note: it will be helpful to draw the transposed matrix A, along with the submatrix partitioning first. Check your implementation using a common matrix transpose operator and matrix comparison using `isequal()`.



Divide this task into two separate functions with the following signatures:

`function [ S1, S2, S3, S4, S5, S6 ] = foo( A )`

Divides the matrix **A** into submatrices.

`function B = bar( S1, S2, S3, S4, S5, S6 )`

Transposes the matrix using the submatrices show above as inputs.

Exercise 2: Generate pseudo-random matrices

Implement a function to generate integer pseudo-random values with the given function prototype: `function R = random( interval, dimensions )`, where the parameter `interval` is a vector containing the boundaries of a closed interval < a, b > of possible generated values (ie `interval = [ a, b ], where b > a`) and the parameter `dimensions` is the dimension of the resulting generated matrix (or vector or scalar) R (ie `dimensions = [ M, N ]`)). This function should be implemented as a wrapper above the `rand()` function, whose outputs should be transformed in a suitable way so that they meet the required range of values. In your implementation, it is forbidden to use the **`randi()` and `randn()`** function. Help: first read the short help: `help rand`.

Exercise 2a: Unit test for your random function

Implement a function to test your `random()` function from the previous exercise. The function signature of the unit test is given by: `function [ A, E, F ] = testRandom()`. Let this function internally implement the following commands:

1) Generates 4x4 matrix **A** containing only **integer** values from the interval < 1, 10 >
2) Generates a 4x1 column vector **b**, containing only integer values in from the interval < 1, 3 >
3) Generates a 4x1 column vector **c**, containing only integer values in from the interval < 3, 6 >
4) Generates a 4x1 column vector **d**, containing only integer values in from the interval < 6, 9 >
5) Multiplies the vectors with matrix A:

   `B = A * b, C = A * c, D = A * d`
6) Use horizontal concatenation to store the resulting vectors **B, C, D** as columns in matrix **F**.
7) Use horizontal concatenation to store the resulting vectors **b, c, d** as columns in matrix **E**.

In the main program, call the function `testRandom()`, multiply the matrix **E** by the matrix **A**, and compare the result with the matrix **F**. The result of the comparison must be a scalar logical value (0 or 1, not a vector or matrix).

Exercise 3: Block matrix multiply

a. Generate six constant matrices $\mathbf{A}_{11}$ … $\mathbf{A}_{22}$, and $\mathbf{B}_{11}$, $\mathbf{B}_{21}$ as shown below:

$$\mathbf{A}_{11}=\begin{pmatrix}1&1&1&1\\1&1&1&1\\1&1&1&1\end{pmatrix},\ \mathbf{A}_{12}=\begin{pmatrix}2&2\\2&2\\2&2\end{pmatrix},\ \mathbf{A}_{21}=\begin{pmatrix}3&3&3&3\\3&3&3&3\end{pmatrix},\ \mathbf{A}_{22}=\begin{pmatrix}4&4\\4&4\end{pmatrix}$$

Define a function `genA()`, that will internally generate and then return all the matrices $\mathbf{A}_{11}$ to $\mathbf{A}_{22.}$ Use the built-in function `ones()` in your implementation. Also do the same for matrices $\mathbf{B}_{11}$ and $\mathbf{B}_{21}$ by defining another function `genB()`.

Let the function prototypes for this exercise be:

```
function [ A11, A12, A21, A22 ] = genA()
function [ B11, B21 ] = genB()
function [ C11, C21 ] =  mulPP( A11, A12, A21, A22, B11, B21 )
```

$$\mathbf{B}_{11}=\begin{pmatrix}5&5&5\\5&5&5\\5&5&5\\5&5&5\end{pmatrix},\mathbf{B}_{21}=\begin{pmatrix}6&6&6\\6&6&6\end{pmatrix}$$

b. Join the matrices using the horizontal and vertical concatenation to form matrices **A** and **B** as defined below:

$$\mathbf{A}^{5\times6}=\begin{pmatrix}\mathbf{A}_{11}^{\ 3\times4}&\mathbf{A}_{12}^{\ 3\times2}\\\mathbf{A}_{21}^{\ 2\times4}&\mathbf{A}_{22}^{\ 2\times2}\end{pmatrix},\ \mathbf{B}^{6\times3}=\begin{pmatrix}\mathbf{B}_{11}^{\ 4\times3}\\\mathbf{B}_{21}^{\ 2\times3}\end{pmatrix}$$

c. Multiply the matrices **A** and **B** in two ways:
   1) Directly using matrix multiplication operation on **A** and **B**. Store result to variable **C**.
   2) Define a new function `mulPP()`, that will take 6 input parameters: submatrices $\mathbf{A}_{ij}$ and $\mathbf{B}_{ij}$, multiply the submatrices per-partes by using multiplication and addition of $\mathbf{A}_{11}$ … $\mathbf{A}_{22}$, and $\mathbf{B}_{11}$, $\mathbf{B}_{21}$, and finally output two submatrices $\mathbf{C}_{11}$ and $\mathbf{C}_{21}$.Use matrix concatenation and store the result to variable **C2**.

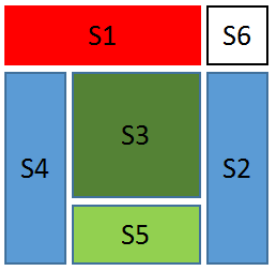
d. Call all the functions from previous tasks and compare the resulting matrices **C** and **C2**. The result of the comparison must a scalar logical value (not a vector or matrix).

Cvičenie 4: Základy maticového počtu

Všetky úlohy vypracujte jedného MATLAB skriptu (m-súboru) s názvom `lab4_XXX.m` (Kde namieste XXX napíšete Vaše priezvisko bez diakritiky). Použite bunkový režim (`%%`) pre oddelenie úloh. Pozor: definície funkcií musia byť na konci m-súboru. **Pri všetkých úlohách je zakázané použitie akýchkoľvek cyklov ako napr. `for/while` ako aj funkcií: `horzcat, vertcat, transpose` – miesto nich použite operátory jazyka MATLAB. Ak pracujete vo dvojiciach, pridajte obe Vaše mená do komentára na prvý riadok odovzdaného súboru.**

Úloha 1: Indexácia matice

Pomocou indexácie prostredníctvom rozsahov rozdeľte maticu **A** rozmerov 4x4 na submatice podľa obrázka. Potom zo submatíc S1 – S6 vyskladajte maticu **B**, ktorá bude transpozíciou matice **A**. Maticu **B** najprv vygenerujte ako nulovú maticu, a pomocou indexácie prostredníctvom rozsahov do nej na správne miesta umiestnite čiastkové matice S1 – S6. Poznámka: je vhodné si najprv nakresliť obrázok transponovanej matice **A** spolu s jej rozdelením na submatice.Overte si správnosť implementácie pomocou bežnej transpozície a porovnania výsledných matíc pomocou `isequal()`.



Rozdeľte si túto úlohu na dve funkcie s nasledujúcimi prototypmi:

`function [ S1, S2, S3, S4, S5, S6 ] = foo( A )`

Rozdelí maticu **A** na submatice.

`function B = bar( S1, S2, S3, S4, S5, S6 )`

Transponuje maticu **A** len s použitím submatíc.

Úloha 2: Generovanie pseudonáhodných matíc a násobenie matíc a vektorov

Implementujte funkciu na generovanie **celočíselných** pseudo-náhodných hodnôt s daným funkčným prototypom: **`function R = random( interval, dimensions )`**, kde parameter `interval` je vektor obsahujúci hranice uzavretého intervalu < a, b > možných generovaných hodnôt (teda `interval = [ a, b ]`, kde `b > a`) a parameter dimensions je rozmer výslednej generovanej matice (alebo vektora, či skalára) R (teda `dimensions = [ M, N ]`)). Táto funkcia má byť implementované ako wrapper nad funkciou **`rand()`**, ktorej výstupy má vhodným spôsobom transformovať tak, aby spĺňali požadovaný rozsah hodnôt. Pri Vašej implementácií je zakázané použiť funkcie **`randi()` a `randn()`**. Pomôcka: najprv si prečítajte krátku nápovedu: **`help rand`**.

Úloha 2a: Unit test pre funkciu random

Implementujte funkciu na testovanie Vašej funkcie `random()` z predchádzajúcej úlohy. Funkčný prototyp funkcie je daný: `function [ A, E, F ] = testRandom()` . Táto funkcia nech vnútorne implementuje nasledujúce príkazy:

1) Vygeneruje maticu **A** rozmerov 4x4 obsahujúcu len celočíselné hodnoty z intervalu < 1, 10 >
2) Vygeneruje stĺpcový vektor **b**, obsahujúci len celočíselné hodnoty z intervalu < 1, 3 >
3) Vygeneruje stĺpcový vektor **c**, obsahujúci len celočíselné hodnoty z intervalu < 3, 6 >
4) Vygeneruje stĺpcový vektor **d**, obsahujúci len celočíselné hodnoty z intervalu < 6, 9 >
5) Vynásobí všetky vektory maticou **A**:
   `B = A * b, C = A * c, D = A * d`
6) Použije operátor horizontálneho spájania matíc na uloženie vektorov **B**, **C**, **D** ako stĺpcov matice **F**.
7) Použije operátor horizontálneho spájania matíc na uloženie vektorov **b**, **c**, **d** ako stĺpcov matice **E**.

V hlavnom programe zavolajte funkciu `testRandom()`, vynásobte maticu **E** maticou **A** a porovnajte výsledok s maticou **F**. Výsledok porovnania musí byť skalárna logická hodnota (0 alebo 1, nie vektor alebo matica).

Úloha 3: Blokové násobenie matíc

a. Vygenerujte 6 konštantných matíc $\mathbf{A}_{11}$ … $\mathbf{A}_{22}$, a $\mathbf{B}_{11}$, $\mathbf{B}_{21}$ podľa nasledujúcich definícii:

$$\mathbf{A}_{11}=\begin{pmatrix}1&1&1&1\\1&1&1&1\\1&1&1&1\end{pmatrix},\ \mathbf{A}_{12}=\begin{pmatrix}2&2\\2&2\\2&2\end{pmatrix},\ \mathbf{A}_{21}=\begin{pmatrix}3&3&3&3\\3&3&3&3\end{pmatrix},\ \mathbf{A}_{22}=\begin{pmatrix}4&4\\4&4\end{pmatrix}$$

Definujte funkciu `genA()`, ktorá interne vygeneruje a vráti ako svoje návratové hodnoty všetky matice $\mathbf{A}_{11}$ až $\mathbf{A}_{22}$ .Pri svojej implementácií využite funkciu `ones()`. To isté spravte pri maticiach $\mathbf{B}_{11}$ a $\mathbf{B}_{21}$ a definujte funkciu `genB()`.
Funkčné prototypy funkcií sú pevne dané:

```
function [ A11, A12, A21, A22 ] = genA()
function [ B11, B21 ] = genB()
function [ C11, C21 ] =  mulPP( A11, A12, A21, A22, B11, B21 )
```

$$\mathbf{B}_{11}=\begin{pmatrix}5&5&5\\5&5&5\\5&5&5\\5&5&5\end{pmatrix},\mathbf{B}_{21}=\begin{pmatrix}6&6&6\\6&6&6\end{pmatrix}$$

b. Spojte matice pomocou operátorov horizontálneho a vertikálneho spájania matíc do matíc **A** a **B** definovaných ako:

$$\mathbf{A}^{5\times6}=\begin{pmatrix}\mathbf{A}_{11}^{\ 3\times4}&\mathbf{A}_{12}^{\ 3\times2}\\\mathbf{A}_{21}^{\ 2\times4}&\mathbf{A}_{22}^{\ 2\times2}\end{pmatrix},\ \mathbf{B}^{6\times3}=\begin{pmatrix}\mathbf{B}_{11}^{\ 4\times3}\\\mathbf{B}_{21}^{\ 2\times3}\end{pmatrix}$$

c. Vynásobte matice **A** a **B** dvoma spôsobmi:
   1) Priamo pomocou operátora násobenia matíc **A** a **B**. Výsledok uložte do premennej **C**.
   2) Definujte funkciu `mulPP()`, ktorá má 6 vstupných parametrov: matice $\mathbf{A}_{ij}$ a $\mathbf{B}_{ij}$, a ktorá vynásobí tieto submatice po častiach – pomocou násobenia a sčítania submatíc $\mathbf{A}_{11}$ … $\mathbf{A}_{22}$, and $\mathbf{B}_{11}$, $\mathbf{B}_{21}$ výsledky uložte do dvoch submatíc $\mathbf{C}_{11}$ a $\mathbf{C}_{21}$. Použite operátor spájania matíc na zostavenie výslednej matice a túto priraďte do premennej **C2**.

d. Zavolajte všetky funkcie definované v predchádzajúcich bodoch a porovnajte výsledné matice **C** a **C2**. Výsledok porovnania musí byť skalárna logická hodnota (0 alebo 1, nie vektor alebo matica).