A0B17MTB – Matlab

Part #11

Miloslav Čapek
miloslav.capek@fel.cvut.cz

Viktor Adler, Filip Kozák, Pavel Valtr

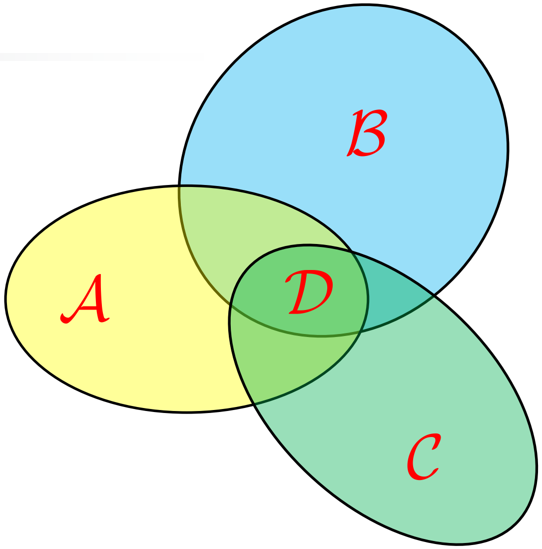
Department of Electromagnetic Field
B2-634, Prague

Learning how to …

Set operations

Sorting

Searching



$$\mathcal{D} = \mathcal{A} \bigcap \mathcal{B} \bigcap \mathcal{C}$$

$$\mathcal{A} \bigcap \mathcal{B} = \left\{ x : x \in \mathcal{A} \wedge x \in B \right\}$$

Department of Electromagnetic Field, CTU FEE, `miloslav.capek@fel.cvut.cz`

Set operations

- there exist following operations (operators) in Matlab applicable to arrays or individual elements
  - arithmetic (part #1)
  - relational (part #4)
  - logical (part #4)
  - set (part #7)
  - bit-wise (help, >> `doc`)

- set operations are applicable to vectors matrices, arrays, cells, strings, tables, ...
  - mutual sizes of these structures are usually not important

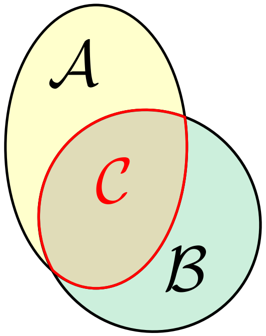
intersection of two sets	`intersect`
union of two sets	`union`
difference of two sets	`setdiff`
exclusive OR of two sets	`setxor`
unique values in a set	`unique`
sorting, row sorting	`sort`, `sortrows`
is the element member of a set?	`ismember`
is the set sorted?	`issorted`

Department of Electromagnetic Field, CTU FEE, miloslav.capek@fel.cvut.cz

Set operations #1

- intersection of sets: `intersect`
    - example: intersection of a matrix and a vector:

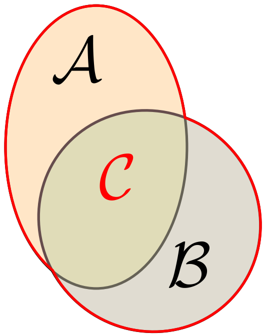
```
>> A = [1 -1; 3 4; 0 2];
>> b = [0 3 -1 5 7];
>> c = intersect(A, b)
% c = [-1; 0; 3]
```



$$\mathcal{C} = \mathcal{A} \cap \mathcal{B}$$

- union of sets: `union`
    - all set operations can be carried out row-wise (in that case the number of columns has to be observed)

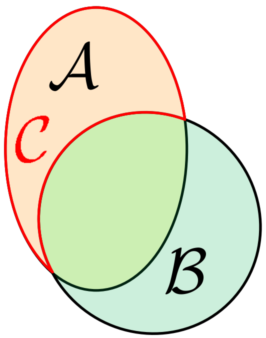
```
>> A = [1 2 3; 4 5 1; 1 7 1];
>> b = [4 5 1];
>> C = union(A, b, 'rows')
% C = [1 2 3; 1 7 1; 4 5 1]
```



$$\mathcal{C} = \mathcal{A} \cup \mathcal{B}$$

Set operations #2

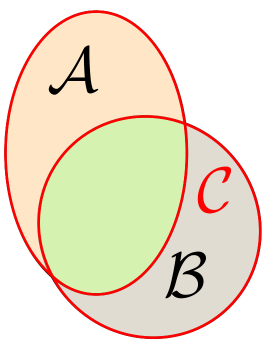
- intersection of a set and complement of another set : `setdiff`
  - all set operations return more than one output parameter - we get the elements as well as the indexes



$$\mathcal{C} = \mathcal{A} \bigcap \mathcal{B}^{\mathrm{C}} = \mathcal{A} \setminus \mathcal{B}$$

```
>> A = [1 1; 3 NaN];
>> B = [2 3; 0 1];
>> [C, ai] = setdiff(A,B)
% C = NaN, ai = 4
% i.e.: C = A(ai)
```

- exclusive intersection (XOR): `setxor`
  - all set operations can be carried out either as '*stable*' (not changing the order of elements) or as '*sorted*' (elements are sorted)



$$\mathcal{C} = \mathcal{A} \oplus \mathcal{B}$$

```
>> A = [5 1 0 4];
>> B = [1 3 5];
>> [C, ia, ib] = setxor(A, B, 'stable')
% C = [0 4 3], ia = [3; 4], ib = [2]
```


intersect
union
setdiff
setxor
unique
sort, sortrows
ismember
issorted

Department of Electromagnetic Field, CTU FEE, miloslav.capek@fel.cvut.cz

Set operations #3

- selection of unique elements of an array : `unique`
  - set operations are also applicable to arrays not (exclusively) containing numbers

$$\begin{pmatrix} c & b & a & c \\ a & c & b & a \\ c & c & d & b \end{pmatrix} \supseteq \begin{pmatrix} a \\ b \\ c \\ d \end{pmatrix}$$

```
>> A = {'Joe', 'Tom', 'Sam'};
>> B = {'Tom', 'John', 'Karl', 'Joe'};
>> C = unique([A B])
% C = {'John', 'Karl', 'Joe', 'Sam', 'Tom'}
```

intersect
union
setdiff
setxor
unique
sort, sortrows
ismember
issorted

- it is possible to combine all above mentioned techniques
  - e.g. row-wise listing of unique elements of a matrix including indexes:

```
>> A = round(rand(10, 3)).*mod(10:-1:1, 3)'
>> [C, ai, ci] = unique(sum(A, 2), 'rows', 'stable')
```

  - Interpret the meaning of the above code? Is the `'rows'` parameter necessary?

Set operations #1

600 s

- consider three vectors **a**, **b**, **c** containing natural numbers $x \in \mathbb{N}$ so that
  - vector **a** contains all primes up to (and including) 1000
  - vector **b** contains all even numbers up to (and including) 1000
  - vector **c** is complement of **b** in the same interval

- find vector **v** so that $\mathbf{v} = \mathbf{a} \cap (\mathbf{b} + \mathbf{c}),\quad \mathbf{b} + \mathbf{c} \equiv \{b_i + c_i\},\quad i \in \{1, 500\}$
  - what elements does **v** contain? $b_{i-1} < b_i < b_{i+1} \;\wedge\; c_{i-1} < c_i < c_{i+1},\; \forall i$

- how many elements are there in **v**?

```
v =

  Columns 1 through 24

     3     7    11    19    23    31    43    47    59    67    71    79

  Columns 25 through 48

   211   223   227   239   251   263   271   283   307   311   331   347

  Columns 49 through 72

   491   499   503   523   547   563   571   587   599   607   619   631

  Columns 73 through 87

   823   827   839   859   863   883   887   907   911   919   947   967
```

Set operations #2

500 s

- estimate the result of following operation (and verify using Matlab):

$$\mathbf{w} = \left(\mathbf{b} \cup \mathbf{c}\right) \setminus \mathbf{a}$$

  - what is specific about elements of the resulting vector **w**?

- with the help of logical indexing and mathematical functions determine how many elements of **w** are divisible by 3

Department of Electromagnetic Field, CTU FEE, miloslav.capek@fel.cvut.cz

Set operations #3

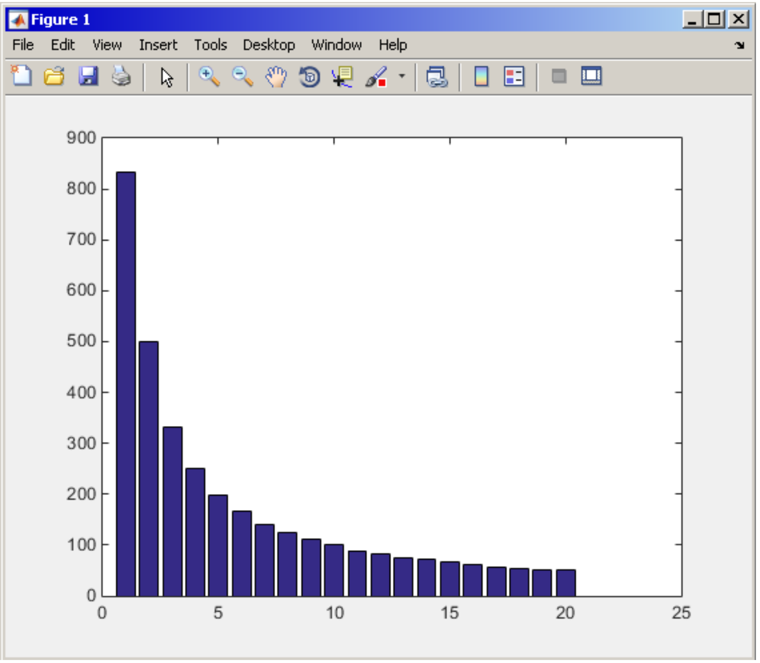
500 s

- write previous exercise as a script:

- modify the script in the way to calculate how many elements of **w** are divisible by numbers 1 to 20
  - use for instance `for` loop to get the result
  - plot the results using `bar` function

Department of Electromagnetic Field, CTU FEE, miloslav.capek@fel.cvut.cz

Set operations #4



Set operations #5

600 s

- Radio relay link operates at frequency of 80 GHz at 20 km distance with 64-QAM modulation
  - phase stability of ±0.5° is required for sufficiently low bit error rate without using synchronization and coding
  - that corresponds to the change of distance between antennas equal to ±5 μm
  - the statistics of link distance with normal distribution containing $1\cdot10^6$ elements can be generated as:

```
L = 20e3; % length of path
deviation = 5e-6; % standard deviation
N = 1e6; % number of trials
% random distances
distances = L + randn(1, N)*deviation;
```

- How many times is the distance `L` contained in the vector `distances`?
- How many unique elements are there in `distances`?
- Can the distribution be considered continuous?

Department of Electromagnetic Field, CTU FEE, miloslav.capek@fel.cvut.cz

Array sorting #1

intersect
union
setdiff
setxor
unique
sort, sortrows
ismember
issorted

- sort array elements
  - column-wise, in ascending order:

```
>> sort(A)
```

  - row-wise, in ascending order :

```
>> sort(A, 2)
```

  - in descending order:

```
>> sort(A, 'descend')
```

  - in descending order, row-wise:

```
>> sort(A, 2, 'descend')
```

- apply the sorting function, to following matrices (for instance):

```
>> A = reshape([magic(3)  magic(3)'], [3 3 2])
>> B = 'for that purpose';
```

Array sorting #2

- function `sortrows` sorts rows of a matrix
  - elements of the rows are not swapped - rows are sorted as blocks

$$\begin{pmatrix} 8 & 1 & 6 \\ 3 & 5 & 7 \\ 4 & 9 & 2 \end{pmatrix}$$

SORT:

$$\left(\begin{array}{c|c|c} 3 & 1 & 2 \\ 4 & 5 & 6 \\ 8 & 9 & 7 \end{array}\right)$$

SORTROWS:

$$\begin{pmatrix} \boxed{3 \quad 5 \quad 7} \\ \boxed{4 \quad 9 \quad 2} \\ \boxed{8 \quad 1 \quad 6} \end{pmatrix}$$

Department of Electromagnetic Field, CTU FEE, miloslav.capek@fel.cvut.cz

`is*` functions related to sets

- function `issorted` returns `true` if array is sorted

- function `ismember(A,B)` tests whether an element of array `B` is also an element of array `A`

```
>> ismember([1 2 3; 4 5 6; 7 8 9], [0 0 1; 2 1 4])
```

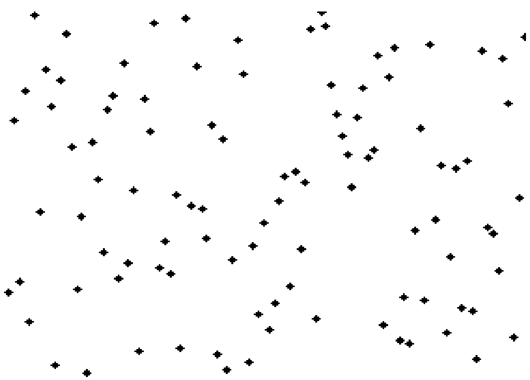
```
>> ismember([1 2 3; 4 5 6; 7 8 9], [0 0 1; 2 1 4])

ans =

     1     1     0
     1     0     0
     0     0     0
```


Department of Electromagnetic Field, CTU FEE, miloslav.capek@fel.cvut.cz

Array sorting

600 s

- try to write your own sorting algorithm `bubbleSort.m`
  - use the *bubble sort* algorithm
  - use the function `issorted` to test whether the resulting array is sorted



wikipedia.org

```
sort(R)
```

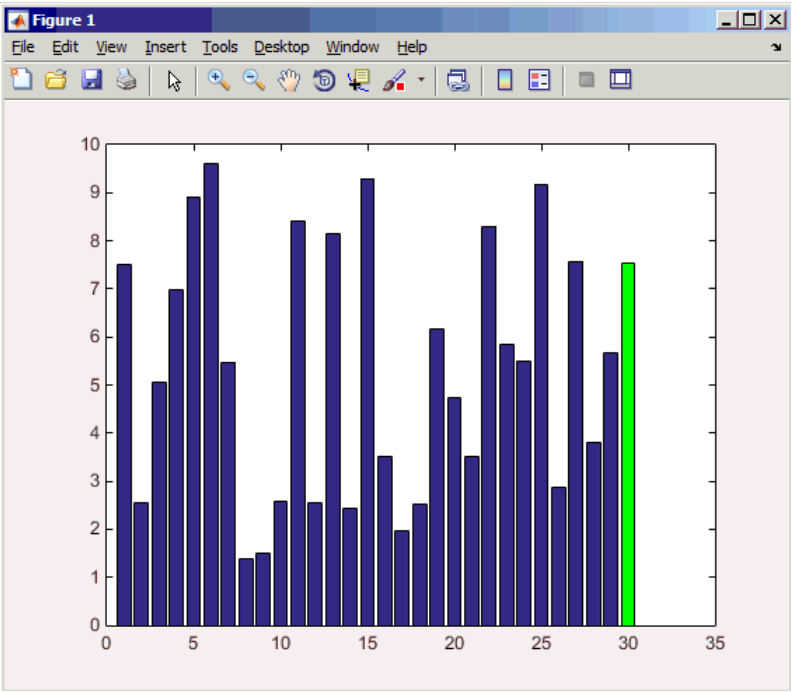
if you wish, you can use the following code inside loops :

```
figure(1);
plot(R,'*','LineWidth',2);
pause(0.01);
```

Array sorting

600 s

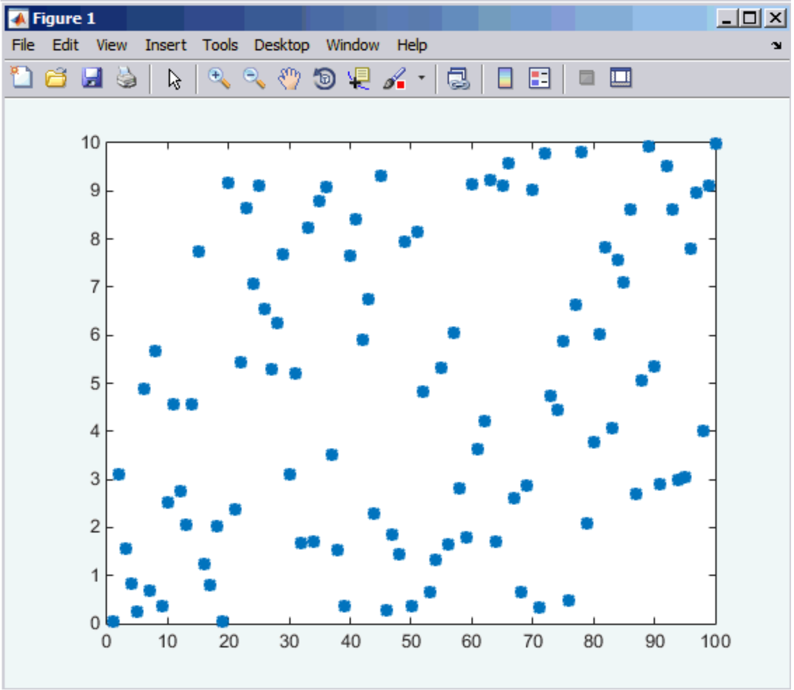
- try to get plot as in the figure using `bar` function:



Array sorting – shaker sort

600 s

- try to write your own sorting algorithm `shakerSort.m`
  - use the *shaker sort* algorithm



Department of Electromagnetic Field, CTU FEE, miloslav.capek@fel.cvut.cz

Searching in an array – `find`

- `find` function is a very useful one!!
- returns positions of non-zero (logical true) elements of a matrix
  - useful for searching in an array of logical values
  - example: find positions of those elements of vector $\mathbf{A} = \begin{pmatrix} \frac{\pi}{2} & \pi & \frac{3}{2}\pi & 2\pi \end{pmatrix}$ fulfilling the condition $\mathbf{A} > \pi$

```
>> A = pi/2*(1:4)
>> find(A > pi)
```

  - compare the above command with `A > pi`. What is the difference?
- function `find` can also search a square matrix etc.
- to find first / last `k` non-zero elements of `X`:

```
>> ind = find(X, k, 'first')
>> ind = find(X, k, 'last')
```

- for more see `>> doc find`

Array searching #1

420 s

- sort the vector $\mathbf{v} = (16 \quad 2 \quad 3 \quad 13 \quad 5 \quad 11 \quad 10 \quad 8 \quad 9 \quad 7 \quad 6 \quad 12 \quad 4 \quad 14 \quad 15 \quad 1)$ in descending order and find the elements of the vector (and their respective positions within the vector) that are divisible by three and at the same time are greater than 10

```
>> v  = reshape(magic(4)', [1 numel(magic(4))])
```

```
v =

    16     2     3    13     5    11    10     8     9     7     6    12     4    14    15     1

v1 =

     0     1     0     0     1     0     0     0     0     0     0     0     0     0     0     0

ans =

    15    12

ans =

     2     5
```

Array searching #2

300 s

- in matrix **w**

```
>> w = (8:-1:2)'*(1:1/2:4).*magic(7)
```

find last 3 values that are smaller than 50

- find out the column and row positions of the values

```
w =

  240.0000  468.0000  768.0000   20.0000  240.0000  532.0000  896.0000
  266.0000  493.5000   98.0000  157.5000  378.0000  661.5000  812.0000
  276.0000   54.0000   96.0000  255.0000  468.0000  735.0000  888.0000
   25.0000  105.0000  160.0000  312.5000  510.0000  630.0000  900.0000
   52.0000   90.0000  192.0000  330.0000  504.0000  616.0000   64.0000
   63.0000  103.5000  192.0000  307.5000  387.0000   31.5000  144.0000
   44.0000   93.0000  160.0000  245.0000   12.0000   77.0000  160.0000
```


Department of Electromagnetic Field, CTU FEE, miloslav.capek@fel.cvut.cz

Application of the `find` function

600 s

- Samples of demodulated signal of a radio receiver can be approximated as :

```
w = 0.6833; t = 1:10; % time
samples = 2.7 + 0.5*(cos(w*t) - sin(w*t) - cos(2*w*t) + sin(2*w*t) ...
   - cos(3*w*t) + 3*sin(3*w*t) + 2*cos(4*w*t) + 4*sin(4*w*t));
plot(samples, '*')
```

- Voltage corresponding to characters are within ±0.5 V tolerance
- Decipher the message!

Voltage [V]	Character
1	a
2	c
3	d
4	g
5	m
6	r
7	s

```
chars = 'acdgmrs'; volts = 1:7;
message = blanks(length(samples));
for iVolt = volts
   logCondition = samples > (iVolt - 0.5) & ...
      samples < (iVolt + 0.5);
   indices = find(logCondition);
   message(indices) = chars(iVolt);
end
disp(message)
```

Function `accumarray` #1

- the function `accumarray` is able to group data with the same index
  - not a very well known function, but an exceptionally useful one
- quite often we deal with a dataset that is organised in the following way:

```
>> ind  = [1 1 2 3 1 2 4 4].';
>> data = [.1 .3 1 -3.1 .2 1.1 10.1 10.2].';
>> Dta1 = accumarray(ind, data)
```

indexes (e.g. measurement number)

values (measured)

```
ind =        data =

     1          0.1
     1          0.3
     2          1.0
     3         -3.1
     1          0.2
     2          1.1
     4         10.1
     4         10.2
```

```
Dta1 =

    0.6000    = 0.1 + 0.3 + 0.2   (ind == 1)
    2.1000    = 1.0 + 1.1         (ind == 2)
   -3.1000    = -3.1              (ind == 3)
   20.3000    = 10.1 + 10.2       (ind == 4)
```

Function `accumarray` #2

- basic operation applicable to data from one 'box' (data with the same index) is summation
- any other function can be applied, however
  - e.g. maximum of a set of elements with the same index
  - we use the `max` function

```
>> Dta2 = accumarray(ind, data, [], @max)
```

```
Dta2 =
   0.3000
   1.1000
  -3.1000
  10.2000
```

  - e.g. listing of all elements with the same index
  - we use so called handle function and `cell` data type

```
>> Dta3 = accumarray(ind, data, [], @(x) {x})
```

```
Dta3 =
  [3x1 double]
  [2x1 double]
  [   -3.1000]
  [2x1 double]
```


Department of Electromagnetic Field, CTU FEE, miloslav.capek@fel.cvut.cz

Function `accumarray` #3

- the function has a wide variety of other features
- it is possible, for instance, to use 2D indexation of results
  - the results are not put in a 1D set of 'boxes' but to a 2D array instead

```
>> ind = [1 1;2 2;1 2;1 3;1 1;3 1];
>> data = [10 22 12 13 1 pi];
>> Dta4 = accumarray(ind, data)
```

```
ind =

   1   1
   2   2
   1   2
   1   3
   1   1
   3   1
```

```
data =

    10
    22
    12
    13
     1
    pi
```

ind == [1 1] 10 + 1 = **11**	ind == [1 2] **12**	ind == [1 3] **13**
ind == [2 1] **0**	ind == [2 2] **22**	ind == [2 3] **0**
ind == [3 1] **pi**	ind == [3 2] **0**	ind == [3 3] **0**

Function accumarray

300 s

- account transfers in CZK, EUR a USD are as follows
  - (CZK ~ 1, EUR ~ 2, USD ~ 3)

- find out account balance in each currency
  - the exchange rate is 26 CZK = 1€, 25 CZK = 1$, find out total balance

$$\begin{pmatrix} 1 & -110 \\ 1 & -140 \\ 2 & -22 \\ 3 & -2 \\ 2 & -34 \\ 1 & -1300 \\ 2 & -15 \\ 1 & -730 \\ 3 & 24 \end{pmatrix}$$

```
>> dta = [1 -110; 1 -140; 2 -22; 3 -2; ...
          2 -34; 1 -1300; 2 -15; 1 -730; 3 24]
>> K   = [1 26 25]
```

Discussed functions

`intersect`	intersection of sets (vectors / matrices)	
`union`	intersection of sets (vectors / matrices)	
`setdiff`	Subtraction of sets (intersection of a set and complement of another set)	
`setxor`	exclusive intersection	
`unique`	selection of unique elements of an array	
`sort`	sort vector/matrix elements	
`sortrows`	sorts rows of a matrix as a whole	
`accumarray`	group data	●
`ismember`	is given element is member of array?	
`issorted`	is array sorted?	
`find`	find elements fulfilling given condition	●

Exercise #1

- measurement of temperature was carried out in the course of 5 days every second clock hour. Data was measured at 3 different sites (A, B, C)

- find out average daily temperature in given week for all 3 sites
  - i.e., get mean value of measurement at the same hour on the same site

- generate the data using `temperature_measurement.m`
  - see the script on the following slide
  - see the variables required

Department of Electromagnetic Field, CTU FEE, miloslav.capek@fel.cvut.cz

Exercise #2

script for data generation

```
clear; close all; clc;
%% allocation
days = 5; hours = 12;
TimeA = zeros(days*hours,1);
TimeB = TimeA;
TimeC = TimeA;
%% creation of time data-set
for kDay = 1:days
    TimeA((hours*(kDay-1)+1):(hours*(kDay-1)+12),1)  = 2*(randperm(12)-1)';
    TimeB((hours*(kDay-1)+1):(hours*(kDay-1)+12),1)  = 2*(randperm(12)-1)';
    TimeC((hours*(kDay-1)+1):(hours*(kDay-1)+12),1)  = 2*(randperm(12)-1)';
end
%% place and tempreture data-sets
PlaceA = abs(abs(TimeA - 11) - 10) + 10 +  5.0*rand(size(TimeA,1),1);
PlaceB = abs(abs(TimeB - 12) - 10) +  5 + 10.0*rand(size(TimeB,1),1);
PlaceC = abs(abs(TimeC - 11) - 11) +  5 +  7.5*rand(size(TimeC,1),1);

%% generating final variables for the example
TimeAndPlace = [TimeA/2+1   ones(size(TimeA,1),1);...
                TimeB/2+1 2*ones(size(TimeA,1),1);...
                TimeC/2+1 3*ones(size(TimeA,1),1)];
MeasuredData  = [PlaceA; PlaceB; PlaceC];

%% plot final data-set
plot(TimeA,PlaceA,'LineWidth',1,'LineStyle','none','Marker','x',...
    'MarkerSize',15); hold on;
plot(TimeB,PlaceB,'LineWidth',1,'LineStyle','none','Marker','*',...
    'MarkerSize',15,'Color','r');
plot(TimeC,PlaceC,'LineWidth',2,'LineStyle','none','Marker','o',...
    'MarkerSize',10,'Color','g');
set(gcf,'Color','w','pos',[50 50 1000 600]); set(gca,'FontSize',15);
xlabel('time','FontSize',15); ylabel('Temperature','FontSize',15);
title('Measured Data'); grid on; legend('Place A','Place B','Place C');
```

and the results …

Exercise #3

- all the data are contained in 2 matrices:
  - `TimeAndPlace (5×3×12,2) = (180,2)`
  - `MeasuredData (5×3×12,1) = (180,1)`

number of days number of measurement sites number of measurements per day

- unfortunately, data in `TimeAndPlace` are intentionally unsorted

INDEXES:	TimeAndPlace =		MeasuredData =	DATA:
tindex = 10, Place = 1	10	1	15.0797	T(10,1) = 15.0797 °C
	4	1	18.9739	
	7	1	19.3836	
	...	...	...	
	12	2	9.9506	
tindex = 6, Place = 2	6	2	19.7588	T(6,2) = 19.7588 °C
	...	...	...	

Department of Electromagnetic Field, CTU FEE, miloslav.capek@fel.cvut.cz

Exercise #4

600 s

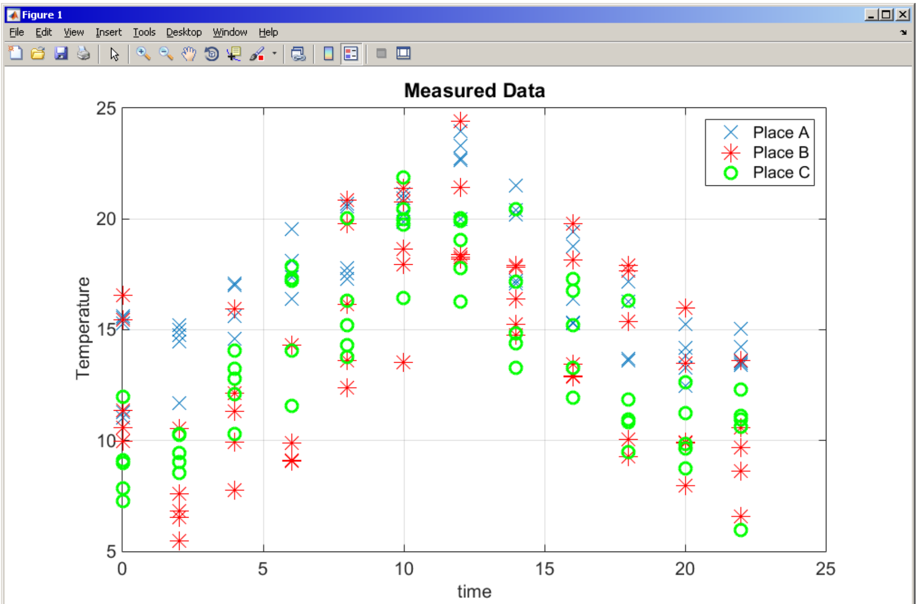
- following holds true
  - Place1 ~ measurement site A
  - Place2 ~ measurement site B
  - Place3 ~ measurement site C
  - measurement hour = 2*(tindex-1)

- now try to place your code in the script to carry out the averaging and plot the data in the existing figure

```
%% PLACE YOUR CODE HERE
%===========================================================================

% ...
% dataA = ...
% dataB = ...
% dataC = ...
%===========================================================================

%% plot the averaged data
plot(0:2:22,dataA,'LineWidth',2,'Color','b','LineStyle','-');
plot(0:2:22,dataB,'LineWidth',2,'Color','r','LineStyle','-');
plot(0:2:22,dataC,'LineWidth',2,'Color','g','LineStyle','-');
```


Department of Electromagnetic Field, CTU FEE, miloslav.capek@fel.cvut.cz

Exercise #5



measured data

measured and averaged data

Thank you!

ver. 9.1 (02/05/2018)

Miloslav Čapek, Pavel Valtr, Viktor Adler
miloslav.capek@fel.cvut.cz

Document created as part of A0B17MTB course.