MATLAB Practical session III: Relational and logical operators

Exercise 1: Relational and logical operators syntax

Part 1: Relational operators

Relational operators compare two operands and determine the validity of a relationship. Refresh your memory on the relational operators available in MATLAB (<, >, <=, >=, ==, ~=) using the following examples:

```
a = 1; b = 2; c = 3; d = 1;
p1 = a == d;     p12 = eq(a,d);   %
p2 = c > b;      p22 = gt(c,b);   %
p3 = c >= b;     p32 = ge(c,b);   %
p4 = a <= c;     p42 = le(a,c);   %
p5 = a ~= d;     p52 = ne(a,d);   %
p6 = a < d;      p62 = lt(a,d);   %
```

Compare the results of the left and right syntaxes. Describe the syntax in words using comments %

Part 2: Logical operators

More complex conditions can be represented by combining relational operations using logical operators. The following logical operators are available in MATLAB: ~, &, |, xor, &&, ||. Use the help function or check your theory book. Compare the results of the left and right syntaxes. Describe the syntax in words using commands %

```
A = [ 5, -3, 0, 0]; B = [2, 4, 0, 5];
z1 = A | B;                      z21 = or(A,B); %
z2 = A & B;                      z22 = and(A,B);%
z3 = ~A;                         z32 = not(A);  %
z4 = ~(A > 4);                                  %
z5 = xor(A,B);                                  %
z6 = (c > b) && (a == d);                       %
z7 = (c < b) && (a <= d);                       %
z8 = (a < d) || (b > a);                        %
z9 = any(A < 5 & B > 8);                        %
```

Note: The short circuit operators (&&, ||) will stop the evaluation of an expression as soon as the results of the entire expression is known. Use the short circuit operators (&&, ||) when comparing single logical values (scalars). Use the non-short circuit operators (&, |) when comparing arrays of logical values.

```
Count = 0; Total = 1000;
(Count ~= 0) && (Total/Count > 80)          % short-circuit
(Count ~= 0) & (Total/Count > 80)           % non-short circuit
(rand(1,3) > rand(1,3)) & ([3 5 9] > [5 2 6])
```

Part 3: Logical indexing

This exercises show techniques of logical-indexing for matrix manipulation. Execute and interpret the results of the following commands:

```
x = [3 16 9 12 -1 0 -12 9 6 1];
y = [3 5 6 1 8 2 9 4 0 7];
a1 = x(x>6);
a2 = y(x<=4);
a3 = x(y<0);
a4 = (x>3)&(x<8);
a5 = x((x<2)|(x>=8));
a6 = y((x<2)|(x>=8));
```

Create the following arrays using logical-indexing

a) Set the positive values of x to zero in a new array x1;
b) Set values of x that are multiples of 3 to 3 (use rem) in a new array x2;
c) Extract the values of x that are >10 in the array x3.

Part 4: Combining relational and logical operators

Create the following statements using relational and logical operators

```
C = randperm(10,10); % random permutations
D = 1:10;
```

a) subtract from D taking 1 for C <= 7 and 0 otherwise
b) ones at positions where 2 <= C < 4
c) ones at positions where C == D or D == 3
d) 1 when ANY of the C elements are larger than 5
e) 1 when ALL D-elements are larger than 2
f) locate all nonzero elements of array C

Part 5: Leap year function

A leap year is a year containing one additional day in order to keep the calendar year synchronized with the astronomical or seasonal year. To determine whether a year is a leap year or not you must check if the year is divisible by 400 or if the year is divisible by 4 and not by 100.

Define a statement 'leapyear' which checks if a year is leap year using logical operators and the `rem` function. Test the statement for different years.

```
year = 2013;
```

Exercise 2: Rugby players

Physiological characteristics of specific individual positions in junior rugby league players.

Step 1: Import data

Import the data form a excel-file called *'RugbyPlayers.xlsx'*. This file contains the physiological characteristics of specific individual positions in junior rugby league players e.g. height, weight, 10, 20 and 40 m sprint, VO2max, agility and vertical jump.

- Compare if the number of columns of the data and the headers are equal using the relational operators and the isequal function. What is the difference?
- Extract the individual positions of the rugby players. Find the column numbers in the header file using string compare (strcmp) and find functions.

Step 2: Questions

To solve the questions use relational and logical operators, string compare and the find functions.

1. Calculate the average Height, Weight and VO2max of the players at different positions. Select the position of players whose height, weight as well as VO2max are above average. The names of the columns are '*height (cm)*', *'Body mass (kg)'* and *'VO2 max'*, respectively.
2. Use the weight en the height to calculate the BMI, with formula weight/height^2, with weight in kg and height in meters. Is there a player position in which the BMI index is above 27?
3. Calculate the average speed in km/h of individual players over the 10m, 20m and 40m sprint and calculate the average speed of all players over the different distances. The data gives the time in seconds over a certain distance. To convert the speeds from m/s to km/h use the following relationship: km/h = 3.6*m/s. Select the position of the players whose speed is lower than the average speed of all the players.
4. Select the position of the players who jump-up at least 25% of their own height in a vertical jump exercise.

Exercise 3: Automatic initial contact and toe-off detection

Step 1: Import data and plot

Import the txt-file called *TreadmillGRF.mot*. This file contains measured GRF during a measurement of a walk on a treadmill. Plot the vertical force of the left and right leg. Plot the time on the x-axis. Add meaningful labels to the axis, legend and the title of the plot. Use a line-width of 1.5 for the curves and a front-size of 14 for the title and labels. Plot the forces (left - black, right – magenta), and change the line-style.

The vertical forces for the left and right leg are indicated with a colheader *'1_ground_force_vy'* and *'ground_force_vy'*, respectively. The colheader indicating the time is *'time'*. Find the column numbers in the data structure using string compare and find functions.

Step 2: Automatic initial contact and toe-off detection

Detect the Initial Contact (IC) and Toe Off (TO) of the gait cycles automatically.

- Create a logical array to detect the moments in which there is contact with the force plate is true, no contact is false. Use a threshold of 0.5 to avoid selecting the noise in the signals.
- Determine the indices of initial contact (IC). These are the indices for which the difference with the next logical number is +1. Use the 'diff' and 'find' commands.
- Determine the indices toe off (TO). These are the indices for which the difference with the next logical number is -1. Check the definition of diff function.
- Plot the IC and TO on the previous graph using marker symbols. Use different symbols to separate IC and TO. Use a different color for the left and right foot. Adapt the legend.

Inspect the figure carefully. Are there any errors in detecting the IC and TO times? Use the interactive handles of the plot.

Put the IC and TO times in a structured array ICTO2 and save the structured array in a mat-file.

Step 3: Compare the IC and TO times with manual selected timings

- Load the saved IC and TO times of the plotting exercise in practical session II, in which you defined the IC and TO manually. This information is saved in a structured array named ICTO
- Draw vertical lines on the instances of IC and TO. Use a different line style for IC and TO separately. Use the same color as the markers of the left and right foot.
- Adapt the xlimit of the figure to display the previous selected gait cycles using xlim function.
- Save the created figure in a png-format and in a fig-format.

Inspect the figure carefully, are the manual selected IC and TO moments corresponding with the automatically determined times? Use the find command and relational and logical operators to check automatically.