Programming and automated signal analysis

Ilse Jonkers

Wouter Aerts, Maarten Afschrift,

Jozefien Burg, Lianne Zevenbergen

Practical session III:
Relational and logical operators

Priority

Precedence	Operation
1 (Highest)	Parenthesis ()
2	Transpose (.'), power (.^), complex conjugate transpose ('), matrix power (^)
3	Unary plus (+), unary minus (-), logical negation (~)
4	Multiplication (.*), right division (./), left division (.\), matrix multiplication (*), matrix right division (/), matrix left division (\)
5	Addition (+), subtraction (-)
6	Colon operator (:)
7	Less than (<), less than or equal to (<=), greater than (>), greater than or equal to (>=), equal to (==), not equal to (~=)
8	Element-wise AND (&)
9	Element-wise OR (|)
10	Short-circuit AND (&&)
11 (Lowest)	Short-circuit OR (||)

EXERCISE 1: RELATIONAL AND LOGICAL OPERATORS

Boolean

- Logical 1 – true
- Logical 0 – false

Relational operators

Operator		Operation
==	eq	Equal to
~=	ne	Not equal to
>	gt	Greater than
>=	ge	Greater than or equal to
<	lt	Less than
<=	le	Less than or equal to

- The relation is always read rom left to right

Relational operators

- Assignment statement ‘=’
  - `A = [10, 12, 8]`
  - `B = [6, 12, 9]`
- Logical relation ‘==’
  - `A == B`
- isequal
  - `isequal(A,B)`

Exercise 1: relational and logical operators

Part 1: relational operators

```
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);
```

Logical operators

Operator		Operation	
~	not	Logical NOT	If *true*, then *false*; if *false*, then *true*
&	and	Logical AND	*True* only if **both** values are *true*
|	or	Logical OR	*True* if **either** value is *true*
xor		Logical exclusive-OR	*True* only if **both** values are different
&&		Logical AND with short-circuit evaluation	*True* only if **both** values are *true*
||		Logical OR with short-circuit evaluation	*True* if **either** value is *true*

- Logical operators are always evaluated from left to right
- Logical operators have a lower priority than relational operators

Short circuit vs. non-short circuit operators

- The short circuit operators (&&, ||) will stop the evaluation of an expression as soon as the results of the entire expression is known.

```
Count = 0; Total = 400;
(Count ~= 0) && (Total/Count > 80)
(Count ~= 0) & (Total/Count > 80)
```

Short circuit vs. non-short circuit operators

- Always use the short circuit operators (&&, ||) when comparing single logical values (scalars)
- Always use the non-short circuit operators (&, |) when comparing arrays of logical values

```
Count = 0; Total = 400;
(Count ~= 0) && (Total/Count > 80)
(rand(1,3) > rand(1,3)) & ([3 5 9] > [5 2 6])
```

Exercise 1: relational and logical operators

Part 2: Logical operators

```
A = [ 5, -3, 0, 0]; B = [2, 4, 0, 5];
a = 1; b = 2; c = 3; d = 1;

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 < c & B > d);
```

Exercise 1: relational and logical operators

Part 3: Logical indexing

```
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));
```

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

Exercise 1: relational and logical operators

Part 4: Combining relational and logical operators

```
C = randperm(10,10); 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

Exercise 1: relational and logical operators

Part 4: Leap year

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.

EXERCISE 2: RUGBY PLAYERS

Exercise 2: Rugby Players

Position	Height (cm)	Body mass (kg)	10m (s)	20m (s)	40m (s)	Agility (s)	Vertical jump (cm)	VO2 max
Prop	184	101	2.14	3.62	6.18	6.37	44	42.2
Hooker	172	70	2.04	3.38	5.89	5.86	47.9	46.9
SecondRow	177	84	2.07	3.47	5.89	6.1	49	45.1
Lock	177	75	2.07	3.49	5.93	5.64	45.2	44.6
Halfback	171	69	1.95	3.52	5.62	6.01	50.4	50.5
FiveEighth	176	72	1.93	3.27	5.71	5.71	48.5	48.3
Centre	177	80	2.02	3.34	5.71	5.89	50.4	47.1
Wing	176	73	2.18	3.49	5.94	5.98	45.4	45.7
Fullback	177	79	2.16	3.39	5.84	5.9	42.8	47.8

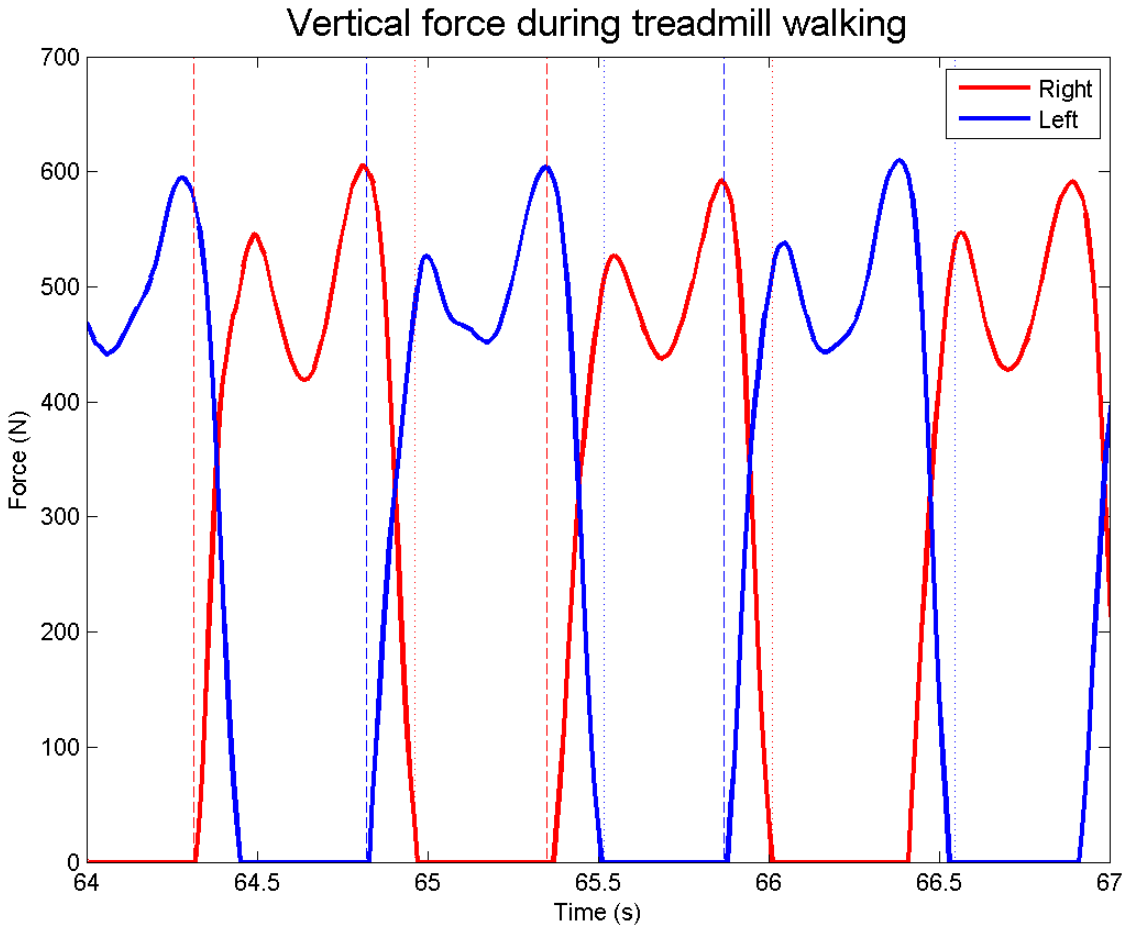
Exercise 2: Rugby players

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

Exercise 3: Automatic initial contact and toe-off detection

- Practical session II: ExerciseICTOGRF



Exercise 3: Automatic initial contact and toe-off detection

'`diff`' function

- `Y = diff(X)` returns a vector of length m-1, if X is a vector of length m. The elements of Y are the differences between adjacent elements of X.

  Y = [X(2)-X(1) X(3)-X(2) ... X(m)-X(m-1)]

Exercise 3: Automatic initial contact and toe-off detection