|  |  |
| --- | --- |
| Biostatica  Door Mark Schrauwen en Alistair Vardy | Matlab Wk2.2 |

Inhoudsopgave

[Versiebeheer 2](#_Toc487553068)

[1 Introductie 3](#_Toc487553069)

[1.1 Eerder behandelde operatoren 3](#_Toc487553070)

[1.2 Nieuwe operatoren 3](#_Toc487553071)

[1.2.1 Antwoord op de vraag 4](#_Toc487553072)

[2 Logische operaties 5](#_Toc487553073)

[2.1 De EN (AND) operator 6](#_Toc487553074)

[2.1.1 De vergelijking tussen true en false 6](#_Toc487553075)

[2.2 De OF (OR) operator 8](#_Toc487553076)

[2.2.1 Voorbeeld van de OR-operaties 8](#_Toc487553077)

[2.3 De NIET (NOT) operator 9](#_Toc487553078)

[2.3.1 Voorbeeld 10](#_Toc487553079)

[2.4 Vragen en opdrachten 11](#_Toc487553080)

[2.5 Antwoorden 12](#_Toc487553081)

[3 Relationele operatoren 13](#_Toc487553082)

[3.1 Output van relationele operatoren 13](#_Toc487553083)

[3.2 Gelijk aan (Equality) operator 13](#_Toc487553084)

[3.2.1 De opbouw op basis van de AND, OR en NOT 13](#_Toc487553085)

[3.2.2 Het verschil tussen de == operator en de = operator 14](#_Toc487553086)

[3.2.3 Andere waardes 14](#_Toc487553087)

[3.3 De NIET-GELIJK-AAN-operator (NOT-EQUAL) 14](#_Toc487553088)

[3.4 Groter dan kleiner dan 15](#_Toc487553089)

[3.4.1 De opbouw op basis van de AND, OR en NOT 16](#_Toc487553090)

[3.5 Element-wise logical operatoren 16](#_Toc487553091)

[3.5.1 Met welk type operator moet worden gewerkt? 17](#_Toc487553092)

[3.5.2 Een ander effect van de short-circuit operator 17](#_Toc487553093)

[3.6 Vragen en opdrachten 18](#_Toc487553094)

[3.7 Antwoorden 19](#_Toc487553095)

[4 Het maken van beslissingen 20](#_Toc487553096)

[4.1 Als-dan (if) 20](#_Toc487553097)

[4.1.1 Wanneer wordt de if-statement uitgevoerd? 21](#_Toc487553098)

[4.2 Als-dan-anders (if-else) 21](#_Toc487553099)

[4.3 Een beslissing met meerdere antwoorden 22](#_Toc487553100)

[4.3.1 Bij geen enkele juiste casus 22](#_Toc487553101)

[4.4 Vragen en opdrachten 23](#_Toc487553102)

[4.5 Antwoorden 26](#_Toc487553103)

[5 Set Path 27](#_Toc487553104)

[5.1 Beginnen bij het begin 27](#_Toc487553105)

[5.2 De oplossing voor ons probleem 29](#_Toc487553106)

[5.3 Conclusie 30](#_Toc487553107)

[6 Bronnen 31](#_Toc487553108)

# Versiebeheer

|  |  |  |  |
| --- | --- | --- | --- |
| Versie | Datum | Beschrijving | Door |
| 0.0 | 11-07-2017 | Eerste versie | Mark Schrauwen |
| 0.1 | 29-08-2017 | Commentaar van Denice Vis verwerkt. | Mark Schrauwen |
|  |  |  |  |

# Introductie

Begin deze week heb je een aantal nieuwe Matlab functies bijgeleerd. Zo weet je nu hoe je een sinus kunt genereren in Matlab. Je weet hoe je een maximum waarde uit een vector haalt en een minimum waarde. *Welke Matlab functies[[1]](#footnote-1) kun je daar ook alweer voor gebruiken?* Ook weet je hoe je de min-tekens uit de elementen van een vector[[2]](#footnote-2) kunt halen. Namelijk met behulp van een abs().

## Eerder behandelde operatoren

In dit deel van de cursus Biostatica – Matlab gaan we kijken naar logische operaties. Dat zijn operaties op nullen en op enen met als resultaat een nul of een één. In lesweek 1.1 hebben we verschillende operaties gezien. Dit is een niet uitputtend overzicht:

**+**

**-**

**\***

**/**

**=**

**^**

**[]**

**.**

Herinner je nog welke operator wat doet? Zo niet kijk dan eens terug de sectie van 1.1 en maak de bijbehorende opgaven om de kennis goed te laten verankeren.

## Nieuwe operatoren

In dit deel gaan we de lijst aan operatoren verder uitbreiden met *logische operatoren*.

*Waarom is dat handig?*

Er zijn verschillende handelingen die we in een programmeertaal willen uitvoeren. De voorgaande handelingen hebben te maken met wiskundige operaties en vector operaties. Echter is het gebruik van een programmeertaal meer dan alleen het uitvoeren van wiskundige en vector handelingen.

Er zijn twee activiteiten die altijd terugkomen als je gaat programmeren ongeacht de programmeertaal:

* Herhalen van iets
* En het maken van beslissingen

Veel handelingen in een programmeertaal moeten worden herhaald. Matlab (en bijna elke andere programmeertaal) heeft allerlei faciliteiten voor het herhalen van bepaalde handelingen. De daadwerkelijk constructies die Matlab heeft om te herhalen worden later behandeld.

Een andere veel voorkomende handeling in een programmeertaal is het maken van beslissingen. Ook hier heeft Matlab (net als bijna elke andere programmeertaal) constructies voor. Deze constructies worden later behandeld. We moeten eerst kennis maken met logische operatoren.

### Antwoord op de vraag

Om antwoord te geven op de vraag aan het begin van deze paragraaf: we gebruiken logische operatoren om in de toekomst in een software programma beslissingen te kunnen maken en om handelingen te kunnen herhalen.

**Zie je een fout? Of heb je een aanbeveling dan horen we dat graag! Stuur dan een e-mail naar** [**mjschrau@hhs.nl**](mailto:mjschrau@hhs.nl) **en wij passen het dan z.s.m. aan.**

# Logische operaties

Het is hopelijk duidelijk geworden waarom er aandacht moet worden besteed aan logische operaties. Als je programma aan het ontwikkelen bent, zal je binnen het programma beslissingen moeten maken en handelingen moeten herhalen.

Een voorbeeld: een student moet op basis van meetdata (krachten in Newton) bepalen of een proefpersoon boven een bepaalde kracht uitkomt. In dat geval moet het programma de kracht weergeven in een figuur. Dan moet er een beslissing worden gemaakt. Hier heb je logische operatoren voor nodig.

Beslissingen en herhalingen worden gebaseerd op *logische operaties* en *relationele operaties*. Laten we beginnen bij het begin: de *logische operaties*.

Het is belangrijk op te merken dat we in de volgende secties telkens uitgaan van twee (binaire) situaties. Situaties die WAAR (TRUE) zijn en situaties die ONWAAR (FALSE) zijn. Meer specifiek, we gaan denken in termen van *stellingen* die WAAR of ONWAAR zijn: ‘Herre Faber is groter, qua lengte, dan Aad Lagerberg’. Deze stelling is trouwens ONWAAR.

Om te herhalen en om te beslissen moeten we stellingen zo ontwerpen dat ze WAAR of ONWAAR zijn.

In Matlab wordt de nul (0) altijd aan gerelateerd aan ONWAAR. En *alles wat ongelijk is aan nul is altijd WAAR*. Vaak wordt dit samengevat als:

Tabel 1: de twee uitgangssituaties

|  |  |
| --- | --- |
| ONWAAR | WAAR |
| 0 | 1 |
| Nul | Één |
| false | true |

**Typ in de Command Window: false**

**Typ in de Command Window: true**

|  |
| --- |
| Wat valt op? Welke waardes zijn gerelateerd aan false en aan true? |

De volgende operatoren, de AND, OR en NOT vormen de basis van de logische operaties van elke digitale omgeving. Onder water werkt alles, echt alles, op basis van deze drie digitale logische operatoren. Ook als je later beslissingen moet maken zal je de AND, OR en NOT wel eens moeten gebruiken.

|  |
| --- |
| **Vanaf dit punt worden standaard de termen true en false gebruikt voor WAAR en ONWAAR.** |

## De EN (AND) operator

Je hebt mogelijk al wel eens gewerkt met de AND operator. Als dat zo is, dan is dit waarschijnlijk een herhaling. Anders zal dit gedeelte duidelijk maken wat de EN-operator gaat doen.

De EN-operator, verder de AND-operator genoemd, heeft twee inputs en één output. Op basis van de twee inputs genereert de AND-operator een output. Hoe de AND-operator dit doet, is te zien in de onderstaande tabel:

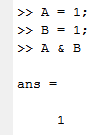
Tabel 2: de AND-operator

|  |  |  |  |
| --- | --- | --- | --- |
|  | Input A | Input B | Output |
| De twee inputs zijn nul | 0 | 0 | 0 |
|  | 0 | 1 | 0 |
|  | 1 | 0 | 0 |
| De twee inputs zijn een | 1 | 1 | 1 |

Tabel 2 laat zien wat de output is als de twee inputs allebei true of ‘1’ zijn. In dat geval is de output ‘1’. In elk ander geval is de output van de AND een nul (‘0’). In Matlab kan de AND operatie op twee manieren gebruikt:

* A & B *(operator vorm)*
* and(A,B) *(functie vorm)*

**Typ in Matlab de volgende code over:**

****

Figuur 1: Voorbeeld van het toepassen van de AND-operator in Matlab

**Pas de waarde van variabele A aan en maak deze nul (A = 0;).**

**Voer opnieuw deze regel uit: A & B**

*Wat valt op?*

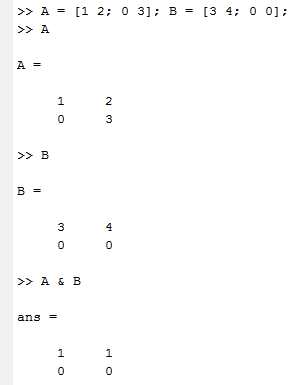
**Typ nu in plaats van ‘A & B’ het volgende ‘and(A,B)’.**

**Herhaal voorgaande actie voor A=1 en B=1;**

|  |
| --- |
| Zitten er verschillen tussen & en and()? **Test dit met Matlab** |

### De vergelijking tussen true en false

Zoals eerder gezegd is de definitie van true in Matlab: *ongelijk aan nul*. Dat dit correct is, laat het volgende voorbeeld zien:



Figuur 2: Voorbeeld van het toepassen van de AND-operator op basis van een matrix.

Figuur 2 laat zien dat de AND-operator werkt op elke element van een matrix. Het laat tevens zien dat volgens de AND-operator alles ongelijk aan nul wordt opgevat als true. Zo wordt de waarde 2 van matrix A gecombineerd in de AND-operator met de waarde 4 van matrix B. Het resultaat (te zien in Figuur 2) is een ‘1’. Dit bewijst dat Matlab een waarde ongelijk aan nul ziet als true.

Dit kun je ook eenvoudiger testen met: 2 & 3

**Voer voorgaande code in en voer deze uit om te controleren of de uitspraak klopt.**

**Test dit nogmaals maar dan met 2 > 3**

Alles ongelijk aan nul wordt door Matlab gezien als true.

## De OF (OR) operator

De OF-operator, verder te noemen de OR-operator, hebben we net zo vaak nodig als de AND-operator. Het gedrag van de OR-operator kenmerkt zich als volgt:

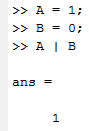
Tabel 3: de OR-operator, dikgedrukt zijn de waardes die anders zijn t.o.v. de AND-operator.

|  |  |  |
| --- | --- | --- |
| Input A | Input B | Output |
| 0 | 0 | 0 |
| 0 | 1 | **1** |
| 1 | 0 | **1** |
| 1 | 1 | 1 |

Zoals in Tabel 3 is te zien heeft de OR-operator een logische naam. Als input A de waarde ‘1’ heeft *of* als de input B de waarde ‘1’ heeft wordt de output (het resultaat van de operator) een ‘1’. In het laatste geval waar A=B=1 is het resultaat van de operatie ook ‘1’.

De OR-operator wordt in Matlab op de volgende manier gebruikt:

* A | B
* or(A,B)



Figuur 3: Voorbeeld van het toepassen van de OR-operator in Matlab. Merk op dat variable B een nul is.

**Typ het voorbeeld in Figuur 3 over en voer het uit**

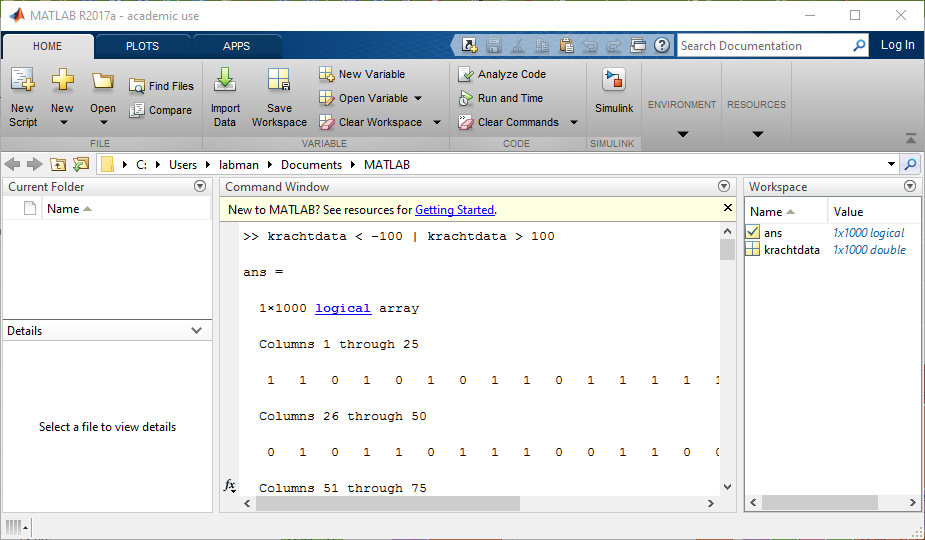
**Gebruik in plaats van de regel ‘A | B’ de regel ‘or(A,B)’. Voer deze regel uit.**

|  |
| --- |
| Wat is het verschil tussen deze notaties?[[3]](#footnote-3) |

**Pas de waardes van de variabele A en B aan en test of de output waardes in Tabel 3 kloppen m.b.v. Matlab**

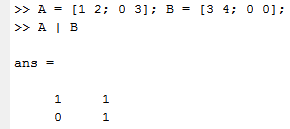
### Voorbeeld van de OR-operaties

Een student krijgt de opdracht om op basis van meetdata te bepalen wanneer waardes van een krachtmeting onder -100 Newton komt en wanneer waardes boven 100 Newton komt. Stel dat de krachtdata in een vector genaamd ‘krachtdata’ staat. De student kan nu gemakkelijk m.b.v. Matlab bepalen welke waardes voldoen aan het bereik (kleiner dan -100 en groter dan 100, zie Figuur 4):



Figuur 4: Een praktisch voorbeeld van de OR-operator. Het filteren van waardes onder en boven bepaalde grenzen.

Het voorbeeld in Figuur 2 gaan we herhalen voor de OR-operator:



Figuur 5: Voorbeeld van het toepassen van de OR-operator in Matlab met als input matrices in plaats van enkele waardes.

## De NIET (NOT) operator

De laatste veel gebruikte logische operator is de NIET-operator, verder te noemen de NOT-operator.

|  |
| --- |
| **Er zijn nog meer logische operatoren maar die zijn voor een Bewegingstechnoloog minder relevant** |

De NOT-operator is een eenvoudige operator die slechts de toestand (TRUE of FALSE) inverteert. Dat wil zeggen dat als de toestand TRUE is maakt de NOT-operator de toestand FALSE. En andersom:

Tabel 4: de NOT-operator.

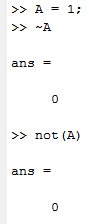
|  |  |
| --- | --- |
| Input A | Output |
| 0 | 1 |
| 1 | 0 |

De NOT-operator wordt in Matlab op de volgende manier gebruikt:

* ~A
* not(A)

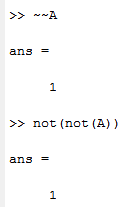
De daadwerkelijke operator is dus de tilde: **~**.

Merk op dat de NOT-operator slechts werkt op één operand en niet op twee operands zoals bij de AND- en OR-operator. In Figuur 6 is het resultaat van de NOT-operator te zien:



Figuur 6: Voorbeeld van het toepassen van de NOT-operator.

Een opvallende eigenschap van de NOT-operator is dat als deze operator twee keer achterelkaar wordt toegepast het resultaat onveranderd is gebleven:



Figuur 7: Voorbeeld van het toepassen van de NOT-operator.

### Voorbeeld

Stel, alleen als een gegeven vector lengte groter dan 0 is, mag je een bepaalde handeling uitvoeren. Hoe test je dat? In Matlab heb je een standaard functie genaamd isempty(). Echter deze functie geeft een *true* terug als een vector leeg is. Terwijl de student graag een stuk code uitvoert als de vector vol is. *Hoe los je dit op?* Door gebruik te maken van een NOT-operator.

## Vragen en opdrachten

1. De AND-operator kan worden gecombineerd met de OR-operator (Ja/Nee)
2. Kan op basis van eenmalig gebruik van de variabele A (=1), B (=0) de AND-operator en de OR-operator het resultaat ‘0’ worden?
3. Maakt het met betrekking tot de vorige vraag uit of je de A en B omdraait?
4. Wat is het resultaat van de volgende combinaties van logische operatoren als A=1 en B=0: or(not(and(A,B)),A). Doe het eerst op papier.
5. Test deze expressie in Matlab
6. Wat is het resultaat van de volgende combinaties van logische operatoren als A=1 en B=0: not(not(and(or(A,B),or(B,A))))? Doe het eerst op papier.
7. Test deze expressie in Matlab
8. Test de expressie van vraag 6 als A=0 en B=0. Doe het eerst op papier en daarna m.b.v. Matlab.
9. Test de expressie van vraag 6 als A=1 en B=1. Doe het eerst op papier en daarna m.b.v. Matlab.
10. Test de expressie van vraag 6 als A=0 en B=1. Doe het eerst op papier en daarna m.b.v. Matlab.
11. Wat is het resultaat van de volgende combinaties van logische operatoren als A=0 en B=1: or(A,and(not(B),not(A)))? Doe het eerst op papier.
12. Test deze expressie in Matlab
13. Geef de functie vorm van de AND-operator
14. Geef het symbool van de OR-operator
15. Geef de functie vorm vande NOT.
16. Geeft het symbool van de NOT-operator.

## Antwoorden

1. Ja dat kan inderdaad. Stel A=1 en B=0: and(or(A,B),A) geeft als resultaat ‘1’. Snap je waarom?
2. De A, B, AND en OR moeten één keer worden gebruikt. Dit zijn dan de mogelijkheden:
   * and(or(A,B),A)
   * and(or(A,B),B)
   * or(and(A,B),A)
   * or(and(A,B),B)

Het resultaat van de voorgaande mogelijkheden is altijd 1. Het maakt bovendien niet uit of je A en B omdraait.

1. Nee
2. Veel van de antwoorden op deze vragen kun je zelf achterhalen m.b.v. Matlab. Dit is de uitwerking:

Or(not(and(1,0))1)

Or(not(0,1)

Or(1,0)

Or = 1

1. Voer het in, in Matlab. Matlab geeft dan het antwoord.
2. 1, Veel van de antwoorden op deze vragen kun je zelf achterhalen m.b.v. Matlab. Dit is de uitwerking:

Not(not(and(or(A,B),or(B,A))))

Not(not(and(or(1,0),or(0,1))))

Not(not(and,or(1,1)))

Not(not(1))

Not(0) = 1

1. Klopt, Voer het in, in Matlab. Matlab geeft dan het antwoord.
2. Voer het in, in Matlab. Matlab geeft dan het antwoord.

Not(not(and(or(A,B),or(B,A))))

Not(not(and(or(0,0),or(0,0))))

Not(not(and,or(0,0)))

Not(not(0))

Not(1) = 0

1. Voer het in, in Matlab. Matlab geeft dan het antwoord.

Not(not(and(or(A,B),or(B,A))))

Not(not(and(or(1,1),or(1,1))))

Not(not(and,or(1,1)))

Not(not(1))

Not(0) = 1

1. Voer het in, in Matlab. Matlab geeft dan het antwoord.

Not(not(and(or(A,B),or(B,A))))

Not(not(and(or(0,1),or(0,1))))

Not(not(and,or(1,1)))

Not(not(1))

Not(0) = 1

1. Voer het in, in Matlab. Matlab geeft dan het antwoord.

Or(A,and(not(B),not(A)))

Or(0,and(not(1),not(0)))

Or(0,and(0,1))

Or(0,0)

= 0

1. Voer het in, in Matlab. Matlab geeft dan het antwoord.
2. and(input1,input2)
3. |
4. not(x)
5. ~ (Alt N voor Macbook users)

# Relationele operatoren

Wat zijn relationele operatoren? Tot nu toe kunnen we logische acties uitvoeren maar nog niet een actie op basis van de stelling: als variabele A groter is dan variabel B moet het programma… Een beslissing op basis groter dan, kleiner dan gelijk aan, etc. gebeurt op basis van relationele operatoren.

Mogelijk denk je over het voorgaande hoofdstuk: “*logische operatoren ga ik toch nooit gebruiken*”. In dit hoofdstuk zal duidelijk worden dat je de logische operatoren sneller nodig hebt dan je erg in hebt. Maar voordat het zover is, worden eerst de relationele operatoren behandeld.

In essentie zijn de relationele operatoren opgebouwd, onder water, op basis van logische handelingen (AND, OR en NOT). In het gebruik zijn de relationele operatoren net zo belangrijk als de logische operatoren. Waarschijnlijk zal je de relationele operatoren zelfs vaker gebruiken dan de logische operatoren.

## Output van relationele operatoren

Net als bij de logische operatoren is de output van de relationele operatoren altijd een ‘true’ of een ‘false’. Maar wat zijn de relationele operatoren? Dat wordt in de volgende paragrafen behandeld.

## Gelijk aan (Equality) operator

De GELIJK-AAN-operator, verder te noemen de EQUALITY-operator of EQ-operator, vergelijkt twee operands met elkaar. Als de operands gelijk zijn aan elkaar is de output true. Zie ook Tabel 5.

Tabel 5: de EQUALITY-operator.

|  |  |  |
| --- | --- | --- |
| Input A | Input B | Output |
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

De EQ-operator wordt in Matlab op de volgende manier gebruikt:

* A == B
* Eq(A,B)

### De opbouw op basis van de AND, OR en NOT

***Deze paragraaf bevat extra informatie en mag worden overgeslagen.***

De stelling eerder in dit hoofdstuk is dat alles is opgebouwd uit AND, OR en NOT operatoren. Dat betekent dat de EQ-operator ook op basis van deze operatoren kan worden opgebouwd. Dat kan worden aangetoond. De EQ-operator als functie van de AND, OR en NOT is:

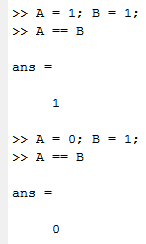
~A & ~B | A & B

### Het verschil tussen de == operator en de = operator

De =-operator (de toekenningsoperator) heb je al vaker gebruikt. Namelijk bij het toekennen van een waarde aan een variabele.

De EQ-operator bestaat uit twee ‘==’ symbolen. Tezamen is dit de EQ-operator.

Het is belangrijk het verschil goed in de gaten te houden.



Figuur 8: het gebruik van de EQUALITY-operator. In het eerste voor beeld is A=B=1. In het tweede voorbeeld is A=0, B=1.

|  |
| --- |
| Wat is het resultaat van de equality operator als A=B=0? Dus A == B? **Test dit met Matlab**.  Snap je waarom? Dit werkt overigens niet voor strings. |

### Andere waardes

Tot nu toe hebben we telkens met ‘0’ en ‘1’ (binaire) waardes gewerkt. De EQ-operator kan net als de logische operatoren worden toegepast op niet binaire waardes (dus 2,3,4,…).

**Typ in Matlab de volgende regel: ‘A=2; B=2; A==B’**

**Druk op Enter**

**Typ in Matlab de volgende regel: ‘A=2; B=45; A==B’**

*Is het resultaat in overeenstemming met je verwachting?*

## De NIET-GELIJK-AAN-operator (NOT-EQUAL)

De NOT-EQUAL operator is de logische tegenhanger van de EQ-operator. Dat wil zeggen dat het resultaat van de NOT-EQUAL-operator, verder te noemen de NE-operator, het geïnverteerde resultaat geeft t.o.v. de EQ-operator:

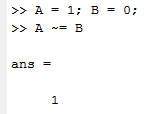
Tabel 6: de NOT-EQUAL-operator. Dikgedrukt de verschillen met de EQ-operator.

|  |  |  |
| --- | --- | --- |
| Input A | Input B | Output |
| 0 | 0 | **0** |
| 0 | 1 | **1** |
| 1 | 0 | **1** |
| 1 | 1 | **0** |

De NE-operator kan, nu we ervaring hebben met de NOT-operator, ook worden gemaakt m.b.v. de NOT-operator en de EQ-operator. Dit wordt als opdracht aan de student gegeven in de vragen.

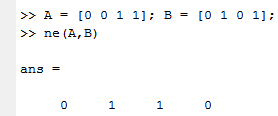
De NE-operator wordt in Matlab op de volgende manier gebruikt:

* A ~= B
* ne(A,B)



Figuur 9: Voorbeeld van het toepassen van de NE-operator.

De NE-operator kan ook op basis van de functie vorm (ne(A,B)) worden toegepast:



Figuur 10: Voorbeeld van het toepassen van de NE-operator. Snap je dit voorbeeld?

## Groter dan kleiner dan

Tijdens de wiskunde lessen van het voortgezet onderwijs heb je kennisgemaakt met de groter-dan, kleiner-dan, kleiner-gelijk-aan en groter-gelijk-aan symbolen en operaties. In Matlab kunnen we ook gebruik maken van deze operaties. In de volgende tabel worden de wiskundige symbolen getoond met de bijbehorende Matlab operatoren

Tabel 7: de overige relationele operatoren in relatie tot hun wiskundige symbolen, haal ze niet door elkaar!

|  |  |  |  |
| --- | --- | --- | --- |
| **Naam** | **Wiskundige symbool** | **Matlab operator** | **Naamgeving** |
| Is gelijk aan |  | == | EQ-operator |
| Groter dan |  | > | GT-operator |
| Kleiner dan |  | < | ST-operator |
| Groter dan gelijk aan |  | >= | GE-operator |
| Kleiner dan gelijk aan |  | <= | SE-operator |

Het gebruik van deze operatoren ga je in de volgende opdrachten oefenen.

### De opbouw op basis van de AND, OR en NOT

***Deze paragraaf bevat extra informatie en mag worden overgeslagen.***

De stelling eerder in dit hoofdstuk is dat alles is opgebouwd uit AND, OR en NOT operatoren. Dat betekent dat de GT-operator ook op basis van deze operatoren kan worden opgebouwd. Dat kan worden aangetoond. De GT -operator (A>B) als functie van de AND, OR en NOT is:

A & ~B

Tabel 8: de GT-operator.

|  |  |  |
| --- | --- | --- |
| Input A | Input B | Output |
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

Voor de GE-operator is dit (met A>=B):

A + ~B

Tabel 9: de GE-operator.

|  |  |  |
| --- | --- | --- |
| Input A | Input B | Output |
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

## Element-wise logical operatoren

Tot nu toe heb je gewerkt met *Element-wise logical* operatoren. Dat zijn operatoren die werken op elk element van een vector (zie ook Figuur 10). Echter is dat niet altijd wat we willen. Vooral niet als we beslissingen maken. Als we beslissingen gaan maken, moet het resultaat altijd een 1 of een 0 (true of false) zijn.

Daarom zijn de short-circuit operatoren voor bedacht. Een short-circuit operator ziet er als volgt uit:

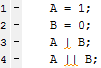
Tabel 10: de overige relationele operatoren in relatie tot hun wiskundige symbolen

|  |  |
| --- | --- |
| **Naam** | **Matlab operator** |
| Element-wise AND | & |
| Short-circuit AND | && |
| Element-wise OR | | |
| Short-circuit OR | || |

Een short-circuit operator heeft dus twee keer het symbool van een *element-wise* operator.

*Maar wat is het verschil in gedrag tussen de twee type operatoren (element-wise vs. Short-circuit)?*

Bij een *short-circuit* logische operator is het resultaat altijd *één resultaat* met twee mogelijkheden true of false. In tegenstelling tot de *element-wise* operator waar het resultaat ook een vector met resultaten kan zijn.



Figuur 11: De element-wise OR-operator vs. de short-circuit OR-operator. Wat is het verschil in resultaat?

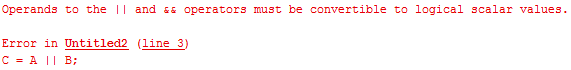
Het resultaat van de code Figuur 11 is in regel 3 en 4 exact hetzelfde namelijk 1. **Test dit zelf uit!** Dus daar zit het verschil hem niet in. Het verschil wordt pas duidelijk als we met vectoren gaan werken:





Figuur 12: De element-wise OR-operator vs. de short-circuit OR-operator. Bij het werken met element-wise OR-operatoren wordt elk element met een OR operator bewerkt.





Figuur 13: De short-circuit versie van de OR-operator geeft een fout melding omdat deze niet met vectoren werkt.

### Met welk type operator moet worden gewerkt?

Dat hangt af van de situatie. In de praktijk wordt de *element-wise* operator zelden tot nooit gebruikt voor herhalingen en beslissingen. De short-circuit operator is veel handiger. Echter kun je in het geval van het testen van een logische situaties (lees: als er geen vectoren in het spel zijn) ook een element-wise operator gebruiken.

Het is ons advies[[4]](#footnote-4) altijd een short-circuit operator (dus een || of een &&) te gebruiken. De reden hiervoor wordt in de volgende paragraaf uitgewerkt

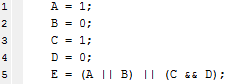
### Een ander effect van de short-circuit operator

***Deze paragraaf bevat extra informatie en mag worden overgeslagen.***

Een ander effect van de short-circuit operator hangt direct samen met de naam van de dit type operator (*short-circuit*).

*Waarom deze naam?*

Deze naam komt af van het gedrag van een short-circuit operator. Dat betekent het volgende: als het resultaat van een operand van een short-circuit het eindresultaat vastlegt hoeft de laatste operand niet nogmaals te worden getest. Een voorbeeld zal helpen:

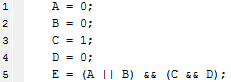


Figuur 14: het resultaat van de expressie in regel 5 wordt opgeslagen in variabele E.

Tot zover heeft het voorbeeld in Figuur 14 nog niet geholpen. Extra uitleg is nodig. Het eerste deel van de expressie in regel 5 (namelijk (A || B)) wordt als eerste getest. Het resultaat is natuurlijk ‘1’. Omdat in regel 5 enkel short-circuit operatoren worden gebruikt is het eindresultaat nu bekend. Immers als 1 wordt geOR-ed met een 0 of 1 is de output 1. Het resultaat wat in variabele E komt te staan is dus bekend.

Short-circuit slaat op het feit dat als het antwoord bekend is de rest van de expressie niet opnieuw getest hoeft te worden. Het voorbeeld in Figuur 15 geeft de andere mogelijkheid weer. De expressie (A || B) is nu nul. We weten dat als A = 0 en B is 1 of 0 dan is de output altijd 0.

In dit geval hoeft (C && D) dus niet meer te worden getest. Immers (A || B) is nul en het eindresultaat is dus ook nul.

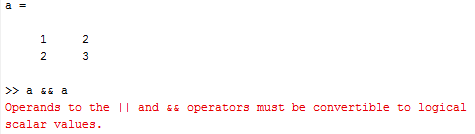


Figuur 15: het resultaat van de expressie in regel 5 wordt opgeslagen in variabele E.

## Vragen en opdrachten

1. Wat is het verschil tussen de =-operator en de ==-operator
2. Wat is de Matlab operator voor de kleiner-dan-gelijk-aan operatie?
3. Wat is het resultaat van deze expressie: 1 <= 0 && 0 <= 1?
4. Geef de Matlab code om te onderzoeken of variabele a niet gelijk is aan variabele b
5. Wat is het verschil tussen een element-wise operator en een short-circuit operator?
6. Wat gebeurt er als je met een short-circuit and-operator twee matrices met elkaar vergelijkt?
7. Waarom kun je het beste een short-circuit operator gebruiken voor het maken van beslissingen?
8. Wat is de output van a & b als a = [1 0 1; 0 1 1; 1 1 1] en b = [ 0 1 1; 0 1 1; 0 1 1;]
9. Wat is de output van ne(a,b) als a = [5 0 8 4 3 2]; en b = [ 6 8 8 0 3 2]; Werk dit eerst uit op papier! Kijk daarna in Matlab of je uitwerking klopt.
10. Als a = [5 0 8 4 3 2]; en b = [ 6 8 8 0 3 2] dan geeft a == b meerdere antwoorden terug (Waar/Niet Waar)

## Antwoorden

1. De =-operator wordt de toekenningsoperator genoemd. Je kent met deze operator een waarde toe aan een variabele (a=4). De ==-operator is een relationele operator (EQ-operator) en daarmee bekijk je of twee variabele dezelfde inhoud bevatten.
2. <=
3. 0
4. a ~= b
5. De element-wise operator werkt op elk element van een vector. De short-circuit operator werkt alleen op enkele variabelen.
6. Dan krijg je een foutmelding: Deze foutmelding is helemaal terecht want je gebruikt deze operator als of het een element-wise operator is.
7. Als je bijvoorbeeld een element-wise and-operator gebruikt voor het maken van een beslissing dan kun je per ongeluk een matrix of vector gebruiken om een beslissing te maken en dat niet alleen slordig het kan ook tot onbetrouwbaar gedrag leiden. Als het eerste element van de vector namelijk nul is dan wordt de if-loop niet uitgevoerd.
8. Deze vraag moet je uit je hoofd kunnen maken: 0 0 1; 0 1 1; 0 1 1
9. Het antwoord is 1 1 0 1 0 0, weet je nog dat ne voor Not Equal stond, als ze dus niet aan elkaar gelijk zijn is het dus een 1!
10. Dit antwoord is waar. Er wordt hier gevraagd of de elementen gelijk zijn aan elkaar niet, er wordt niet gebruikt gemaakt van een short circuit maar van de EQ operator. Het antwoord wat terug wordt gegeven in matlab is 0 0 1 0 1 1

# Het maken van beslissingen

Hoe ben jij hier gekomen? Door te beslissen dat jij verder wilde lezen. Zonder dat je het door hebt, maak je voortdurend beslissingen. Dat gaan we ook doen m.b.v. Matlab. *Waarom zou je beslissingen moeten maken m.b.v. Matlab?*



Figuur 16: als de vrachtwagen groter is dan 11 ft stop!

Het zou zo kunnen zijn dat je een bepaald stuk code op basis van een waarde wel of niet zou willen uitvoeren. Stel je voor dat bij het gebruik van de lichaamslengte in een programma je een stuk code hebt die de werkelijke lichaamslengte van een proefpersoon nodig heeft. Als de betreffende variabele dan nog niet is gevuld met een waarde (hij is dan nul), kan het programma niet verder werken. Op dat moment zou het handig zijn dat jij, als gebruiker van een programma code, wordt gevraagd om de lichaamslengte alsnog in te vullen. Het programma (jouw code) heeft dan een beslissing gemaakt.

Voorgaande situatie zouden we kunnen samenvatten als: *als de variabele lichaamslengte gelijk is aan nul, vraag dan de gebruiker om deze variabele een naam te geven. Ga anders door met het programma.*

*Waar in het voorgaande voorbeeld zit een relationele operator?*

Onderstreept in de vorige gedeelte zijn de sleutelwoorden tot het maken van een beslissing in Matlab: *als-dan.*

## Als-dan (if)

Om in Matlab beslissingen te maken heeft Matlab de if-statement bedacht, zie Figuur 17.



Figuur 17: Voorbeeld van de opzet van een if-statement, voorbeeld code (t.b.v. de uitleg).

Op de plek waar het woord *beslissing* staat moet een beslissing worden gemaakt. De if-statement begint met het woordje if gevolgd door een *logische-expressie* die wordt getest. Inmiddels weet je dat je daar *logische* en *relationele operatoren* voor kunt gebruiken.

De if-statement wordt beëindigd met het woordje: *end*. Tussen de *beslissing* en *end* kan een gebruiker, jij dus, nog meer code zetten die moet worden uitgevoerd (zie Figuur 18).



Figuur 18: Voorbeeld van de opzet van een if-statement volledig werkend.

### Wanneer wordt de if-statement uitgevoerd?

De if-statement wordt niet zomaar uitgevoerd. De logische-expressie (*beslissing)* moet eerst worden getest.

*Hoe doen we dat?*

Met behulp van logische en relationele operatoren. We weten inmiddels dat het resultaat van een dergelijke operaties altijd een nul of een één als resultaat hebben (respectievelijke een false of een true).

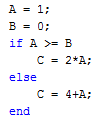
|  |
| --- |
| *Wanneer wordt de if-statement uitgevoerd?*  Antwoord: als de logische-expressie ***true*** is. |

*Wat is het resultaat van de bewerking in Figuur 18?[[5]](#footnote-5)*

Nu is het zo dat als de logische-expressie in Figuur 18 ***false*** is de code in de if-statement niet wordt uitgevoerd. Echter in de praktijk willen we ook een bepaalt gedrag vertonen als de logische-expressie false is. Daar heeft Matlab iets voor bedacht.

## Als-dan-anders (if-else)

Als de logische-expressie false is en we willen een stuk code uitvoeren dan kan met de aanvulling zoals te zien in Figuur 19 ook deze situatie worden uitgevoerd.



Figuur 19: Voorbeeld van de opzet van een if-else-statement

We noemen een dergelijk statement een if-else-statement. Een if-else-statement bestaat dus uit twee delen.

Als de logische expressie true is dan wordt C = 2\*A; uitgevoerd. Als de logische expressie false is dan wordt C = 4+A; uitgevoerd. Slechts één van de twee situaties wordt uitgevoerd.

## Een beslissing met meerdere antwoorden

|  |
| --- |
| Figuur 20: Voorbeeld van een switch case |

Tot nu toe hebben we telkens binaire beslissingen gezien. We doen iets wel of we doen iets niet. Er zijn echter nog meer mogelijkheden. Een expressie kan ook niet-binaire resultaten geven. Bij een optelsom van twee gehele getallen kleiner dan 100 is het eindresultaat altijd kleiner dan 198. Er zijn dan 199 antwoorden mogelijk (van 0 tot 198).

Een ander voorbeeld. Als een gebruiker een keuze moet maken in de toolstrip van Matlab kan hij kiezen uit: HOME, PLOTS, APPS, etc. Er zijn bij een beslissing dus vaak meerdere antwoorden mogelijk. Matlab heeft hier een switch-statement voor bedacht (zie Figuur 20).

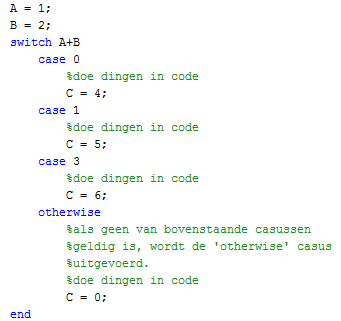
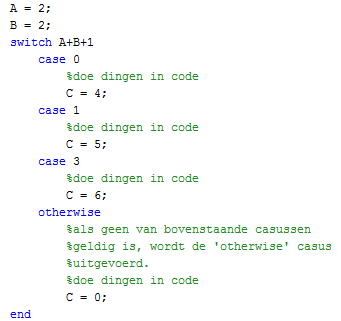
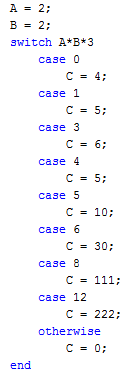
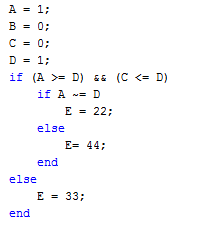
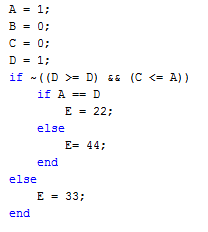
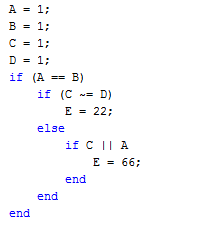
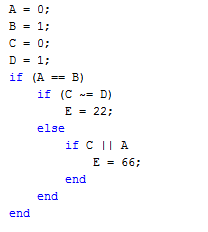
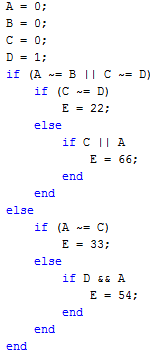
In de switch-statement wordt de expressie (in dit geval in Figuur 20 is de expressie A+B) uitgevoerd. Het resultaat van de expressie in Figuur 20 is natuurlijk 1. In dat geval wordt de code in ***case 1*** (in het Nederlands wordt dit een *casus* genoemd) uitgevoerd. Dat betekent dat de variabele C gelijk wordt aan 5.

### Bij geen enkele juiste casus

Als er geen juist casus is voor de expressie dan is er een standaard casus die wordt gekenmerkt door het woordje *otherwise*. Als geen enkele casus hetzelfde is als de expressie wordt de otherwise casus uitgevoerd.

|  |
| --- |
| Merk op: dat een switch-statement kan worden opgebouwd uit verschillende if-else-statements. |

## Vragen en opdrachten

1. Wat is na het uitvoeren van onderstaande code de waarde in variabele C?
2. Wat is na het uitvoeren van onderstaande code de waarde in variabele C?
3. Wat is na het uitvoeren van onderstaande code de waarde in variabele C?
4. Wat is na het uitvoeren van onderstaande code de waarde in variabele E?
5. Wat is na het uitvoeren van onderstaande code de waarde in variabele E?
6. Wat is na het uitvoeren van onderstaande code de waarde in variabele E?
7. Wat is nu na het uitvoeren van onderstaande code de waarde in variabele E?
8. Wat is nu na het uitvoeren van onderstaande code de waarde in variabele E?
9. De if statement wordt uitgevoerd als de logische-expressie false is (juist/onjuist)

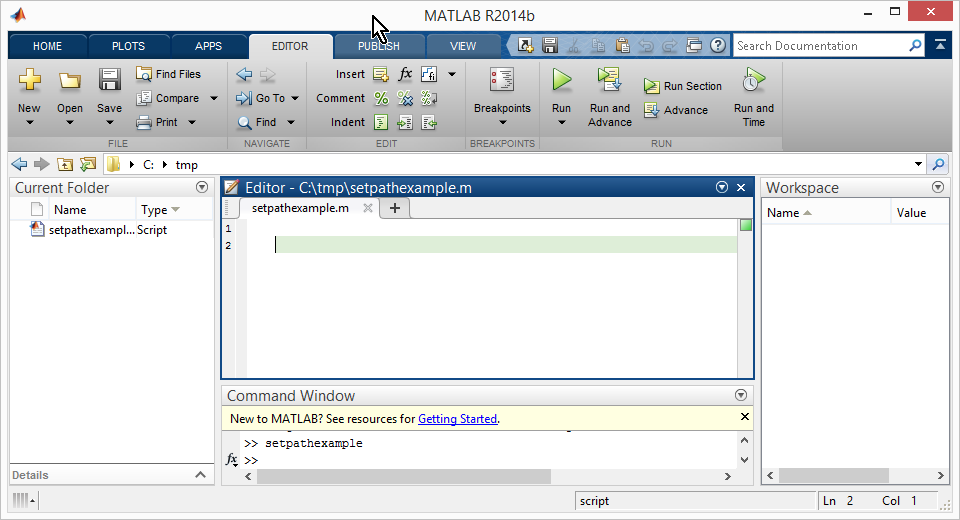
## Antwoorden

1. C = 6.
2. C = 0 via de otherwise-case.
3. C = 0 via de otherwise-case
4. E = 33
5. E = 44
6. E = 66
7. Dit is een strikvraag. De if-statement wordt namelijk niet uitgevoerd.
8. E = 22
9. Deze stelling is onjuist, de statement wordt uitgevoerd als de expressie true is.

# Set Path

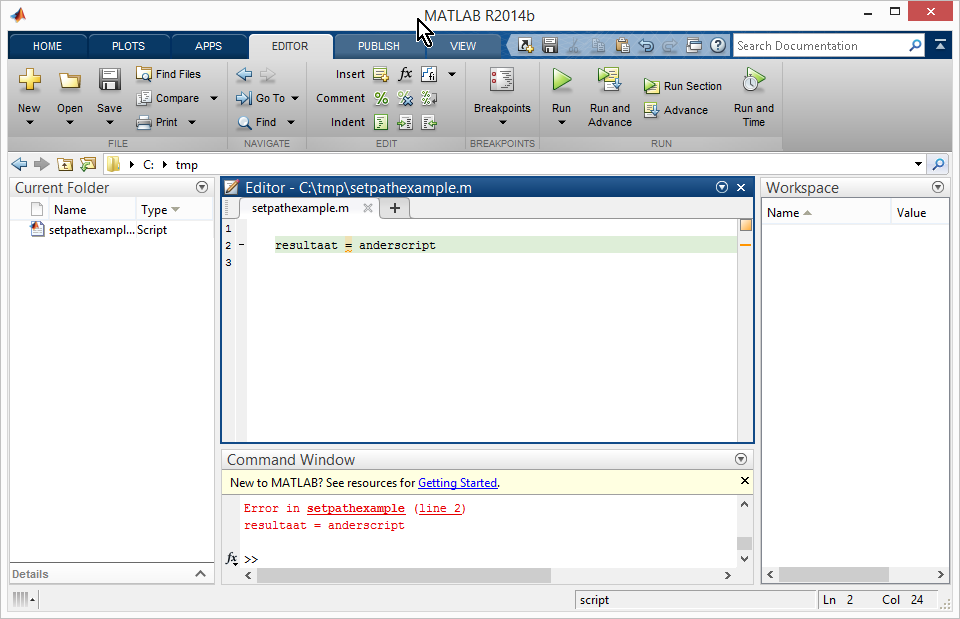
Inmiddels weet je wat functies zijn en heb je er al mee gewerkt. Dit korte hoofdstuk behandeld een veel voorkomend probleem in het gebruik van Matlab. Namelijk het probleem dat *zogenaamd* bepaalde bestanden niet worden gevonden, terwijl jij zeker weet dat de bestanden op je H-schijf of USB-stick staan.

## Beginnen bij het begin



Figuur 21: Onze start situatie: hoeveel bestanden staan er in de Current Folder?

In Figuur 21 is onze start situatie te zien. In de Current Folder staat enkel ons huidig geopende script. We breiden het script uit en het script roept een ander script aan met de naam *anderscript* (zie Figuur 22).

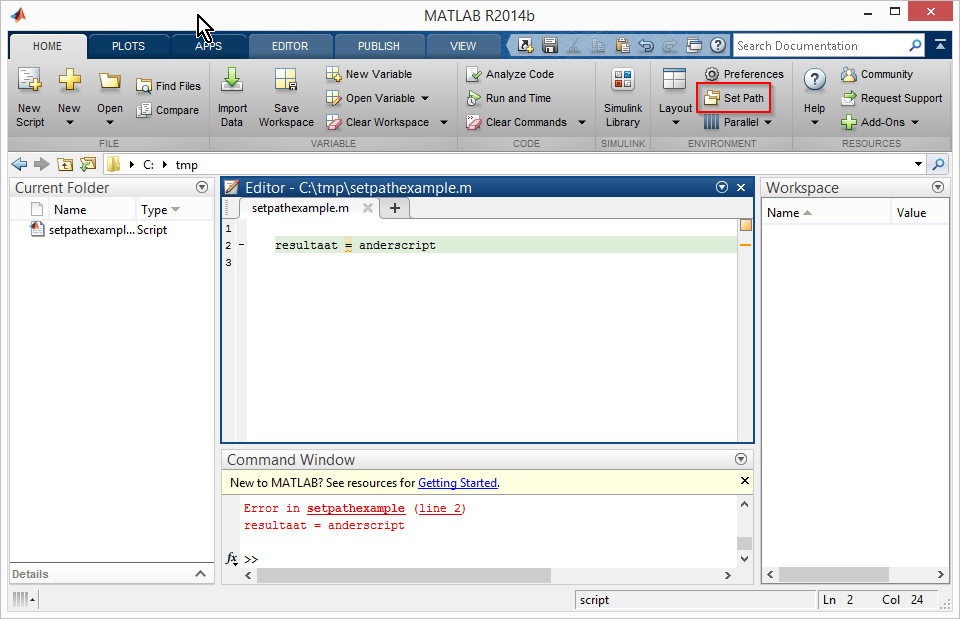


Figuur 22: Het script roept nu een ander script aan (met de naam ‘anderscript’).

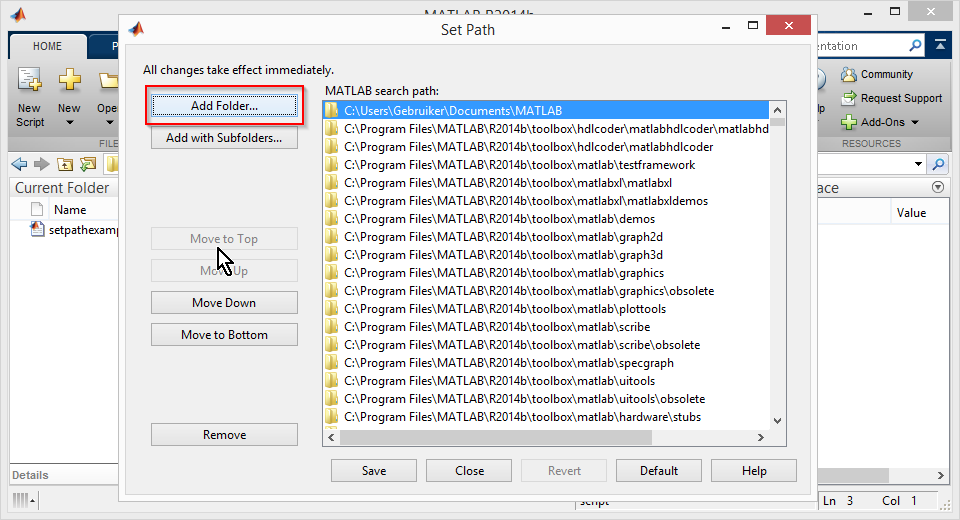
Het aanroepen van een ander script gaat mis. Dat komt omdat Matlab niet weet waar het script met de naam *anderscript* moet worden gezocht. De oplossing voor dit probleem is *Set Path.*

|  |
| --- |
| Volg vanaf dit punt de beschrijvingen van de afbeeldingen |

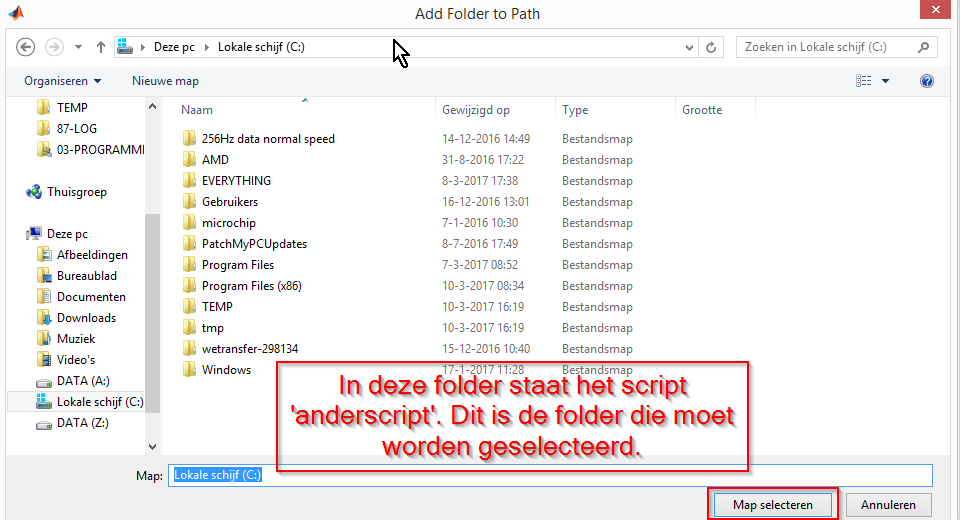
## De oplossing voor ons probleem



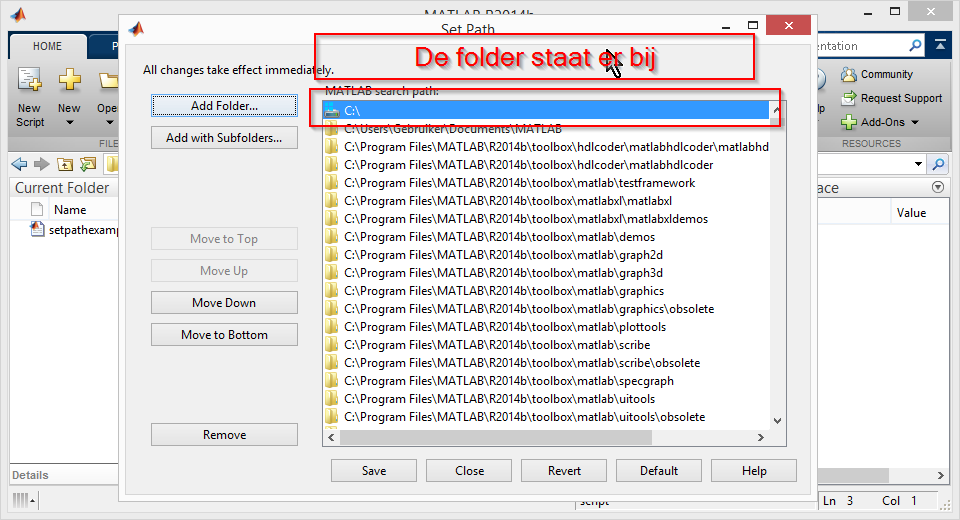
Figuur 23: De Set Path optie staat in het tabblad HOME.



Figuur 24: Het script roept nu een ander script aan (met de naam ‘anderscript’).



Figuur 25: Het script roept nu een ander script aan (met de naam ‘anderscript’).



Figuur 26: Het script roept nu een ander script aan (met de naam ‘anderscript’).

Nu moet op **save** worden gedrukt.

Nu kan het script veilig worden uitgevoerd.

## Conclusie

De conclusie is dat Matlab moet weten waar het script staat. Als Matlab dit niet weet gaat het niet zoeken. Jij moet Matlab wijzen naar de juiste plek. Dat doe je met Set Path.

Er zijn geen opdrachten van hoofdstuk 5.

# Bronnen

* <https://nl.mathworks.com/help/matlab/logical-operations.html>
* <https://nl.mathworks.com/help/matlab/ref/and.html>
* <https://nl.mathworks.com/help/matlab/ref/or.html>

1. Voortaan als je ergens het woord functies ziet staan dan bedoelen wij altijd Matlab functies. [↑](#footnote-ref-1)
2. Vaak kan op een plek waar ‘vector’ is geschreven ook ‘variabele’ worden geschreven. Zoals in dit geval. [↑](#footnote-ref-2)
3. Er is geen verschil. [↑](#footnote-ref-3)
4. Van Mark Schrauwen en Alistair Vardy [↑](#footnote-ref-4)
5. C=2 [↑](#footnote-ref-5)